HLT/EMNLP 2005

Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing

Proceedings of the Conference

6-8 October 2005
Vancouver, British Columbia, Canada

Production and Manufactured by
Omnipress Inc.
Post Office Box 7214
Madison, WI 53707-7214

The conference organizers are grateful to the following sponsors for their generous support.

Silver Sponsor:

Google

Bronze Sponsors:

ELDA

Microsoft

SHARP

XEROX Research Centre Europe

Sponsor of Best Student Paper Award:

IBM

This year's HLT/EMNLP conference is co-sponsored by ***ACL SIGDAT***.

Copyright © 2005 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
3 Landmark Center
East Stroudsburg, PA 18301
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

Preface: General Chair

In 2005, the Human Language Technology Conference (HLT) and the Conference on Empirical Methods in Natural Language Processing (EMNLP) were held together as a joint conference for the first time. The conference was co-sponsored by the organization traditionally behind HLT, the Human Language Technology Advisory Board, and the organization traditionally behind EMNLP, SIGDAT: The Association for Computational Linguistics (ACL) Special Interest Group on linguistic data and corpus-based approaches to natural-language processing. The joint conference was held in Vancouver, B.C., Canada on October 6–8, co-located with the 2005 Document Understanding Conference (DUC) and the 9th International Workshop on Parsing Technologies (IWPT).

In the HLT tradition, the conference especially encouraged submissions involving synergistic combinations of language technologies from the sometimes disjoint areas of natural-language processing, speech processing, and information retrieval. To encourage such cross-fertilization, each of the major chair positions were filled by three people, one from each of these research areas.

First, I would like to thank the Program Chairs, **Chris Brew**, **Lee-Feng Chien**, and **Katrin Kirchhoff**, for handling the unexpectedly large number of submissions under a very tight schedule and putting together an excellent program for the conference. Please see their preface for further information on the submissions, the program committee, and the conference program.

Priscilla Rasmussen deserves our enduring gratitude for agreeing to serve as a remote Local Arrangements Chair, and gracefully handling the multitude of responsibilities that this important position requires.

Joyce Chai did an excellent job as Publications Chair and managing the myriad of details required to assemble this proceedings in the small amount of time allotted for this important step. Thanks also go to **Chen Zhang** and **Shaolin Qu** for helping with the proceedings and to **Jason Eisner** and **Philipp Koehn** for making the publication software available and providing many good suggestions.

Donna Byron, **Anand Venkataraman**, and **Dell Zhang** served as Demonstrations Chairs and carefully reviewed 31 proposals to select 20 interesting demos that were a great addition to the conference program.

David Elworthy and **Marius Pasca** served in the important role of Sponsorship and Exhibits Chairs and helped raise important corporate financial support for the conference. Thanks are also due to our corporate sponsors (listed on the previous page) for their gracious support.

Srinivas Bangalore, **Zak Shafran**, and **Hsin-Min Wang** served as Publicity Chairs and provided important support in advertising the conference to the NLP, speech, and IR communities.

Anoop Sarkar and **Fred Popowich** served as Local Preparation and Student Volunteer Coordinators, providing important local support in Vancouver and assembling and managing a team of student volunteers that provided important services at the conference. The students volunteers themselves also deserve our gratitude.

Yuk Wah Wong, **Razvan Bunescu**, **Ruifang Ge**, and **Rohit Kate** dedicated significant effort as

Webmasters, putting together and constantly updating the conference web site.

Graeme Hirst provided important support and advice as Chair of the HLT Board, particularly in the site selection and initial formation of the conference committee. The members of the HLT Board, **Karen Kukich**, **Donna Harman**, **Mary Harper**, **Julia Hirschberg**, **Sanjeev Khudanpur**, **Joseph Olive**, **John Prange**, **Drago Radev**, and **Ellen Riloff**, also provided important support and advice.

Ken Church also provided important support and advice as chair of SIGDAT in the initial formation of the conference committee and continuing advice on conference organization.

Also thanks to **Donna Harman** for organizing the co-located DUC meeting and **Harry Bunt**, **Rob Malouf** and **Alon Lavie** for organizing the co-located IWPT meeting.

Finally, I would to thank all of the authors, demo presenters, and conference attendees for helping to make the first joint HLT/EMNLP meeting a successful and engaging scientific venue!

Raymond J. Mooney
HLT/EMNLP-05 General Chair
August 24, 2005

Preface: Program Co-chairs

It is our pleasure to welcome you to HLT/EMNLP 2005 in the beautiful city of Vancouver. For the third time, HLT is being held in combination with a conference sponsored by an ACL organization, thus continuing the tradition of bringing together researchers from three different communities: natural language processing, information retrieval, and speech processing. During the last few years, these fields have experienced a growing trend towards interaction across their traditional boundaries, as evidenced by the exchange of approaches and methodologies, and the development of large-scale systems integrating speech and language processing as well as information retrieval components.

We hope that this conference will further encourage this trend. In order to facilitate the interaction between researchers from different fields, all papers have been organized into a single track rather than two or three different tracks. We are also pleased to welcome three invited speakers whose work spans several areas in the HLT/EMNLP field: Ellen Vorhees, Larry Hunter, and Sanjeev Khudanpur. We would like to thank them again for accepting our invitation and for their exciting and stimulating contributions to our program.

The joint organization of HLT and EMNLP generated an unusually large load of papers. A total of 402 submissions were received, of which 127 were accepted, resulting in an acceptance rate of 31.6%. We would like to thank our technical chairs, who did an excellent job at selecting the program committee and managed to handle the large number of submissions efficiently and on time. Our thanks also go to the program committee members for their expert reviews. We are particularly grateful to those PC members who were willing to take on additional reviews beyond their original assignments.

For the demonstrations track, thirty-one submissions were received, twenty of which were accepted. Donna Byron, Anand Venkataramanan and Dell Zhang did a superb job at managing the demo submissions and reviews, and we are looking forward to a very interesting session.

We are please to announce that, for the first time, a prize for the best student paper will be awarded at this year's conference. We are especially grateful to IBM for sponsoring this award – educating future generations of researchers in our community is of prime importance, and the public acknowledgment of students' research achievements is a significant contribution towards this goal.

Finally, we would like to thank our general chair, Ray Mooney, for his help and guidance, and all organizers, PC members, technical chairs, authors, and attendees for their efforts and contributions. We wish you a pleasant time at HLT/EMNLP 2005!

Chris Brew, Lee-Feng Chien, and Katrin Kirchhoff
Program Co-chairs
August 24, 2005

Conference Organizers

General Chair

Raymond J. Mooney, The University of Texas at Austin

Program Chairs

Chris Brew, The Ohio State University
Lee-Feng Chien, Academia Sinica
Katrin Kirchhoff, University of Washington

Demonstrations Chairs

Donna Byron, The Ohio State University
Anand Venkataraman, SRI International
Dell Zhang, Birkbeck, University of London

Publications Chair

Joyce Chai, Michigan State University

Publicity Chairs

Srinivas Bangalore, AT&T Labs - Research
Zak Shafran, Johns Hopkins University
Hsin-Min Wang, Academia Sinica

Sponsorship and Exhibits Chairs

Marius Pasca, Google
David Elworthy, Google

Local Arrangements Chair

Priscilla Rasmussen, Assoc. for Computational Linguistics

Local Preparation and Student Volunteer Coordinators

Fred Popowich, Simon Fraser University
Anoop Sarkar, Simon Fraser University

Webmasters

Yuk Wah Wong, The University of Texas at Austin
Razvan Bunescu, The University of Texas at Austin
Ruifang Ge, The University of Texas at Austin
Rohit Kate, The University of Texas at Austin

Program Committee

Chairs

Chris Brew, The Ohio State University
Lee-Feng Chien, Academia Sinica
Katrin Kirchhoff, University of Washington

Area Chairs

Ricardo Baeza-Yates, University of Chile
Regina Barzilay, Massachusetts Institute of Technology
Jennifer Chu-Carroll, IBM T. J. Watson Research Center
Pascale Fung, Hong Kong University of Science and Technology
Timothy Hazen, Massachusetts Institute of Technology
Rebecca Hwa, University of Pittsburgh
Frank Keller, University of Edinburgh
Elizabeth D. Liddy, University of Syracuse
Dan Melamed, New York University
Helen Meng, Chinese University of Hong Kong
Mark-Jan Nederhof, University of Groningen
Hwee Tou Ng, National University of Singapore
Dan Roth, University of Illinois at Urbana/Champaign
Murat Saraclar, AT&T Labs - Research
Simone Teufel, University of Cambridge
Wayne Ward, University of Colorado
Janyce Wiebe, University of Pittsburgh
Cheng Xiang Zhai, University of Illinois at Urbana/Champaign
Ming Zhou, Microsoft Research Asia

Program Committee Members

Alex Acero (Microsoft Research), Gilles Adda (LIMSI/CNRS, France), Lars Ahrenberg (Linkoping University), Shlomo Argamon (Illinois Institute of Technology)

Michiel Bacchiani (IBM), Ricardo Baeza-Yates (ICREA-UPF & CWR/DCC-Univ. de Chile), Srinivas Bangalore (AT&T Labs - Research), Regina Barzilay (Massachusetts Institute of Technology), Frederic Bechet (LIA, University of Avignon, France), Jerome Bellegarda (Apple Computer), Dan Bikel (IBM Research), Patrick Blackburn (INRIA Lorraine, France), Johan Bos (University of Edinburgh), Antal van den Bosch (Tilburg University, Netherlands), Herve Bourlard (IDIAP Research Institute), Chris Brew (The Ohio State University), Ralf Brown (Carnegie Mellon University), Bill Byrne (The Johns Hopkins University), Donna Byron (The Ohio State University)

Jamie Callan (Carnegie Mellon University), Claire Cardie (Cornell University), Rolf Carlson (Royal Institute of Technology, Sweden), Xavier Carreras Perez (Universitat Politècnica de Catalunya), John Carroll (University of Sussex), Joyce Chai (Michigan State University), Soumen Chakrabarti

(IIT Bombay, India), Ciprian Chelba (Microsoft Research), Francine Chen (Palo Alto Research Center), John Chen (Columbia University), Stanley Chen (IBM T.J. Watson Research Center), David Chiang (University of Maryland), Lee-Feng Chien (Academia Sinica), Jennifer Chu-Carroll (IBM T. J. Watson Research Center), Grace Chung (Corporation For National Research Initiatives), Ken Church (Microsoft), Stephen Clark (Oxford University), Charlie Clarke (University of Waterloo), Michael Collins (Massachusetts Institute of Technology), Ann Copestake (University of Cambridge), Max Copperman, Mark Core (ISI, University of Southern California), Stephen Cox (University of East Anglia), Krzysztof Czuba (IBM T.J. Watson Research Center)

Walter Daelemans (University of Antwerp, Belgium), Ido Dagan (Bar Ilan University, Israel), Renato DeMori (University of Avignon), Mona Diab (Columbia University), Anne Diekema (Syracuse University), Barbara DiEugenio (University of Illinois, Chicago), Shona Douglas (AT&T Labs - Research), John Dowding (National Aeronautics and Space Administration), Amit Dubey (University of Edinburgh), Susan Dumais (Microsoft Research)

Michael Elhadad (Ben-Gurion University of the Negev), Noemie Elhadad (Columbia University), Hakan Erdogan (Sabanci University), Katrin Erk (Saarland University)

David Ferrucci (IBM T.J. Watson Research Center), Radu Florian (IBM Research), Eric Fosler-Lussier (The Ohio State University), George Foster (National Research Council, Canada), Pascale Fung (HLTC/HKUST), Sadaoki Furui (Tokyo Insitute of Technology)

Jianfeng Gao (Microsoft Research Asia), Fred Gey (University of California, Berkeley), Dan Gildea (University of Rochester), Roxana Girju (University of Illinois at Urbana/Champaign), Jade Goldstein (U.S. Department of Defense), Julio Gonzalo (UNED, Spain), Mark Greenwood (University of Sheffield), Greg Grefenstette (CEA, France), Ling Guan (Ryerson University, Toronto), Curry Guinn (University of North Carolina at Wilmington)

Nizar Habash (Columbia University), Udo Hahn (Freiburg University), Thomas Hain (University of Sheffield), Dilek Hakkani-Tur (AT&T Labs - Research), John Hale (Michigan State University), Hans van Halteren (Radboud University Nijmegen), Sanda Harabagiu (University of Texas, Dallas), Mary Harper (Purdue University), Mark Hasegawa-Johnson (University of Illinois, Urbana/Champaign), T. J. Hazen (Massachusetts Institute of Technology), James Henderson (Université de Genève), Julia Hirschberg (Columbia University), Graeme Hirst (University of Toronto, Canada), Julia Hockenmaier (University of Pennsylvania), Thomas Hofmann (Brown University), Ed Hovy (ISI, University of Southern California), David Hull (Clairvoyance Corp.), Rebecca Hwa (University of Pittsburgh)

Rukmini Iyer (Nuance Communications)

Donghong Ji (Institute for Infocomm Research, Singapore), Rong Jin (Michigan State University), Michael Johnston (AT&T Labs - Research), Pamela Jordan (University of Pittsburgh)

Min-Yen Kan (National University of Singapore), Tatsuya Kawahara (Kyoto University), Andy Kehler (University of California, San Diego), Frank Keller (University of Edinburgh), Stephan Kepser (Eberhard Karls Universität Tübingen, Germany), Katrin Kirchhoff (University of Washington), Dan Klein (University of California, Berkeley), Kevin Knight (ISI, University of Southern California), Philipp Koehn (University of Edinburgh), Geert-Jan Kruijff (Universität des Saarlandes, Saarbrücken, Germany), Jonas Kuhn (The University of Texas at Austin), Roland Kuhn (NRC Institute for Information Technology), Shankar Kumar (Johns Hopkins University), Sadao Kurohashi (University of Tokyo, Japan)

Mirella Lapata (University of Edinburgh, UK), Alon Lavie (Carnegie Mellon University), Lin-Shan Lee (National Taiwan University), Oliver Lemon (Edinburgh University), Anton Leuski (University of Southern California), Roger Levy (Stanford University), Mingjing Li (Microsoft Research Asia), Xin Li (University of Illinois at Urbana/Champaign), Elizabeth D. Liddy (Syracuse University), Chin-Yew Lin (ISI, University of Southern California), Dekang Lin (Google), Ting Liu (Harbin Institute of Technology, China), Yang Liu (International Computer Science Institute. Berkeley), Karen Livescu (Massachusetts Institute of Technology), Xiaofei Lu (The Ohio State University), Yajuan Lu (Microsoft Research Asia)

Lidia Mangu (IBM T.J. Watson Research Center), Inderjeet Mani (Georgetown University), Christopher (Manning Stanford University), Daniel Marcu (ISI, University of Southern California), Katja Markert (University of Leeds, UK), Yuval Marom (Monash University), Lluís Màrquez i Villodre (Universitat Politècnica de Catalunya, Spain), Yuji Matsumoto (Nara Institute of Science and Technology, Japan), Andrew McCallum (University of Massachusetts), Diana McCarthy (University of Sussex, UK), Kathleen (McKeown Columbia University), Dan Melamed (New York University), Chris Mellish (University of Aberdeen), Helen Meng (Chinese University of Hong Kong), Rada Mihalcea (University of North Texas), Dan Moldovan (University of Texas, Dallas), Christof Monz (University of Maryland), Bob Moore (Microsoft Research), Tatsunori Mori (Yokohama National University), Sung Hyon Myaeng (Information and Communications University, Korea)

Shri Narayanan (University of Southern California), Mark-Jan Nederhof (University of Groningen), Hermann Ney (RWTH Aachen University), Hwee Tou Ng (National University of Singapore), Vincent Ng (University of Texas at Dallas), Grace Ngai (Hong Kong Polytechnic University), Jian-Yun Nie (University of Montreal), Cheng Niu (Cymfony Corp.), Eric Nyberg (Carnegie Mellon University)

Doug Oard (University of Maryland), Franz Och (Google), Kemal Oflazer (Sabanci University, Turkey), Els den Os (Radboud University Nijmegen), Miles Osborne (University of Edinburgh), Douglas O'Shaughnessy (INRS-Telecommunications)

Martha Palmer (University of Pennsylvania), Shimei Pan (IBM T. J. Watson Research Center), Patrick Pantel (ISI, University of Southern California), Kishore Papineni (IBM Research), Cecile Paris (CSIRO, Australia), Marius Pasca (Google), Bryan Pellom (University of Colorado), Gerald Penn (University of Toronto, Canada), Fernando Pereira (University of Pennsylvania), Michael Picheny (IBM T. J. Watson Research Center), Joe Picone (Mississippi State University), Roberto Pieraccini (IBM), Massimo Poesio (University of Essex), Fred Popowich (Simon Fraser University), John Prager (IBM Reserch), Harry Printz (Agile TV Corporation), Stephen Pulman (Oxford University), Vasin Punyakanok (University of Illinois at Urbana/Champaign)

Dragomir Radev (University of Michigan, Ann Arbor), Bhuvana Ramabhadran (IBM), Owen Rambow (Columbia University), Jason Rennie (Massachusetts Institute of Technology), Philip Resnik (University of Maryland), Christian Retoré (Université Bordeaux 1, France), Giuseppe Riccardi (AT&T Labs - Research), Steve Richardson (Microsoft Research), Frank Richter (Eberhard Karls Universität Tübingen, Germany), Stefan Riezler (Palo Alto Research Center), German Rigau (Universitat Politècnica de Catalunya, Spain), Ellen Riloff (University of Utah), Hae-Chang Rim (Korea University), Brian Roark (Oregon Health and Sciences University), Steve Robertson (Microsoft Research), Dan Roth (University of Illinois at Urbana/Champaign), Salim Roukos (IBM Research), Alex Rudnicky (Carnegie Mellon University)

Yoshinori Sagisaka (ATR/Waseda University), Mark Sanderson (University of Sheffield), Murat Saraclar (AT&T Labs - Research), Hinrich Schuetze (University of Stuttgart, Germany), Sabine Schulte im Walde (Universität des Saarlandes, Saarbrücken, Germany), Fabrizio Sebastiani (Italian National Council of Research, Italy), Frank Seide (Microsoft Research Asia), Stephanie Seneff (Massachusetts Institute of Technology), Jimi Shanahan (Clairvoyance Corp), Libin Shen (University of Pennsylvania), Ronnie Smith (East Carolina University), Frank Soong (Microsoft Research Asia), Karen Sparck Jones (University of Cambridge), Richard Sproat (University of Illinois, Urbana/Champaign), Dave Stallard (BBN Technologies), Mark Steedman (University of Edinburgh), Volker Steinbiss (Accipio Consulting), Amanda Stent (Stony Brook University), Suzanne Stevenson (University of Toronto), Michael Strube (EML Research), Tomek Strzalkowski (University at Albany), Keh-Yih Su (Behavior Design Corporation), Eiichiro (Sumita ATR), Stan Szpakowicz (University of Ottawa)

Joel Tetreault (University of Rochester), Simone Teufel (University of Cambridge), Christoph Tillmann (IBM Research), Tassos Tombros (Queen Mary, University of London), Kristina Toutanova (Stanford University), Gokhan Tur (AT&T Labs - Research)

Takehito Utsuro (Kyoto University)

Andreas Vlachos (Cambridge University), Stephan Vogel (Carnegie Mellon University), Ellen Voorhees (National Institute of Standards and Technology), Sarel van Vuuren (University of Colorado)

Chao Wang (Massachusetts Institute of Technology), Wei Wang (University of North Carolina at Chapel Hill), Ye-Yi Wang (Microsoft Research), Wayne Ward (University of Colorado), Taro Watanabe (NTT Communication Science Lab), Andy Way (Dublin City University), Bonnie Webber (University of Edinburgh), Ralph Weischedel (BBN Technologies), Jan Wiebe (University of Pittsburgh), Hugh Williams (Microsoft Corporation), Florian Wolf (University of Cambridge), Dekai Wu (Hong Kong University of Science and Technology), Xiaoyun Wu (Yahoo)

Fei Xia (IBM Research)

Scott Wentau Yih (Microsoft Research), Steve Young (University of Cambridge)

Hugo Zaragoza (Microsoft Research, Cambridge), Luke Zettlemoyer (Massachusetts Institute of Technology), Cheng Xiang Zhai (University of Illinois at Urbana/Champaign), Ming Zhou (Microsoft Research Asia), Dav Zimak (University of Illinois at Urbana/Champaign)

Table of Contents

Conference Program

Thursday, October 6, 2005

8:45–9:00	Opening Session
9:00–10:00	Invited Talk: Larry Hunter
10:00-10:30	Break
	Session 1A: Anaphora and Information Structure
10:30–10:55	*Improving LSA-based Summarization with Anaphora Resolution* Josef Steinberger, Mijail Kabadjov, Massimo Poesio and Olivia Sanchez-Graillet
10:55–11:20	*Data-driven Approaches for Information Structure Identification* Oana Postolache, Ivana Kruijff-Korbayova and Geert-Jan Kruijff
11:20–11:45	*Using Semantic Relations to Refine Coreference Decisions* Heng Ji, David Westbrook and Ralph Grishman
11:45–12:10	*On Coreference Resolution Performance Metrics* Xiaoqiang Luo
	Session 1B: Error Detection and Correction
10:30–10:55	*Improving Multilingual Summarization: Using Redundancy in the Input to Correct MT errors* Advaith Siddharthan and Kathleen McKeown
10:55–11:20	*Error Detection Using Linguistic Features* Yongmei Shi and Lina Zhou
11:20–11:45	*Semantic Similarity for Detecting Recognition Errors in Automatic Speech Transcripts* Diana Inkpen and Alain Désilets
11:45–12:10	*Redundancy-based Correction of Automatically Extracted Facts* Roman Yangarber and Lauri Jokipii

Friday, October 7, 2005 (continued)

Improving LSA-based Summarization with Anaphora Resolution

Josef Steinberger
University of West Bohemia
Univerzitni 22, Pilsen 30614,
Czech Republic
jstein@kiv.zcu.cz

Mijail A. Kabadjov
University of Essex
Wivenhoe Park, Colchester CO4 3SQ,
United Kingdom
malexa@essex.ac.uk

Massimo Poesio
University of Essex
Wivenhoe Park, Colchester CO4 3SQ,
United Kingdom
poesio@essex.ac.uk

Olivia Sanchez-Graillet
University of Essex
Wivenhoe Park, Colchester CO4 3SQ,
United Kingdom
osanch@essex.ac.uk

Abstract

We propose an approach to summarization exploiting both lexical information and the output of an automatic anaphoric resolver, and using Singular Value Decomposition (SVD) to identify the main terms. We demonstrate that adding anaphoric information results in significant performance improvements over a previously developed system, in which only lexical terms are used as the input to SVD. However, we also show that how anaphoric information is used is crucial: whereas using this information to add new terms does result in improved performance, simple substitution makes the performance worse.

1 Introduction

Many approaches to summarization can be very broadly characterized as TERM-BASED: they attempt to identify the main 'topics,' which generally are TERMS, and then to extract from the document the most important information about these terms (Hovy and Lin, 1997). These approaches can be divided again very broadly in 'lexical' approaches, among which we would include LSA-based approaches, and 'coreference-based' approaches . Lexical approaches to term-based summarization use lexical relations to identify central terms (Barzilay and Elhadad, 1997; Gong and Liu, 2002); coreference- (or anaphora-) based approaches (Baldwin and Morton, 1998; Boguraev and Kennedy, 1999; Azzam et al., 1999; Bergler et al., 2003; Stuckardt, 2003) identify these terms by running a coreference- or anaphoric resolver over the text.[1] We are not aware, however, of any attempt to use both lexical and anaphoric information to identify the main terms. In addition, to our knowledge no authors have convincingly demonstrated that feeding anaphoric information to a summarizer significantly improves the performance of a summarizer using a standard evaluation procedure (a reference corpus and baseline, and widely accepted evaluation measures).

In this paper we compare two sentence extraction-based summarizers. Both use Latent Semantic Analysis (LSA) (Landauer, 1997) to identify the main terms of a text for summarization; however, the first system (Steinberger and Jezek, 2004), discussed in Section 2, only uses lexical information to identify the main topics, whereas the second system exploits both lexical and anaphoric information. This second system uses an existing anaphora resolution system to resolve anaphoric expressions, GUITAR (Poesio and Kabadjov, 2004); but, crucially, two different ways of using this information for summarization were tested. (Section 3.) Both summarizers were tested over the CAST corpus (Orasan et al., 2003), as discussed in Section 4, and sig-

[1]The terms 'anaphora resolution' and 'coreference resolution' have been variously defined (Stuckardt, 2003), but the latter term is generally used to refer to the coreference task as defined in MUC and ACE. We use the term 'anaphora resolution' to refer to the task of identifying successive mentions of the same discourse entity, realized via any type of noun phrase (proper noun, definite description, or pronoun), and whether such discourse entities 'refer' to objects in the world or not.

nificant improvements were observed over both the baseline CAST system and our previous LSA-based summarizer.

2 An LSA-based Summarizer Using Lexical Information Only

LSA (Landauer, 1997) is a technique for extracting the ‘hidden’ dimensions of the semantic representation of terms, sentences, or documents, on the basis of their contextual use. It is a very powerful technique already used for NLP applications such as information retrieval (Berry et al., 1995) and text segmentation (Choi et al., 2001) and, more recently, multi- and single-document summarization.

The approach to using LSA in text summarization we followed in this paper was proposed in (Gong and Liu, 2002). Gong and Liu propose to start by creating a term by sentences matrix $A = [A_1, A_2, \ldots, A_n]$, where each column vector A_i represents the weighted term-frequency vector of sentence i in the document under consideration. If there are a total of m terms and n sentences in the document, then we will have an $m \times n$ matrix A for the document. The next step is to apply Singular Value Decomposition (SVD) to matrix A. Given an $m \times n$ matrix A, the SVD of A is defined as:

(1) $A = U\Sigma V^T$

where $U = [u_{ij}]$ is an $m \times n$ column-orthonormal matrix whose columns are called left singular vectors, $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is an $n \times n$ diagonal matrix, whose diagonal elements are non-negative singular values sorted in descending order, and $V = [v_{ij}]$ is an $n \times n$ orthonormal matrix, whose columns are called right singular vectors.

From a mathematical point of view, applying SVD to a matrix derives a mapping between the m-dimensional space spawned by the weighted term-frequency vectors and the r-dimensional singular vector space. From a NLP perspective, what the SVD does is to derive the *latent semantic structure* of the document represented by matrix A: a breakdown of the original document into r linearly-independent base vectors (‘topics’). Each term and sentence from the document is jointly indexed by these ‘topics’.

A unique SVD feature is that it is capable of capturing and modelling interrelationships among terms so that it can semantically cluster terms and sentences. Furthermore, as demonstrated in (Berry et al., 1995), if a word combination pattern is salient and recurring in document, this pattern will be captured and represented by one of the singular vectors. The magnitude of the corresponding singular value indicates the importance degree of this pattern within the document. Any sentences containing this word combination pattern will be projected along this singular vector, and the sentence that best represents this pattern will have the largest index value with this vector. As each particular word combination pattern describes a certain topic in the document, each singular vector can be viewed as representing a salient topic of the document, and the magnitude of its corresponding singular value represents the degree of importance of the salient topic.

The summarization method proposed by Gong and Liu (2002) should now be easy to understand. The matrix V^T describes the importance degree of each ’implicit topic’ in each sentence: the summarization process simply chooses the most informative sentence for each term. In other words, the kth sentence chosen is the one with the largest index value in the kth right singular vector in matrix V^T.

The summarization method proposed by Gong and Liu has some disadvantages as well, the main of which is that it is necessary to use the same number of dimensions as is the number of sentences we want to choose for a summary. However, the higher the number of dimensions of reduced space is, the less significant topic we take into a summary. In order to remedy this problem, we (Steinberger and Jezek, 2004) proposed the following modifications to Gong and Liu’s summarization method. After computing the SVD of a term by sentences matrix, we compute the length of each sentence vector in matrix V. This is to favour the index values in the matrix V that correspond to the highest singular values (the most significant topics). Formally:

(2) $s_k = \sqrt{\sum_{i=1}^{r} v_{k,i}^2 \cdot \sigma_i^2}$,

where s_k is the length of the vector of k’th sentence in the modified latent vector space, and its significance score for summarization too. The level of dimensionality reduction (r) is essentially learned from the data. Finally, we put into the summary the sentences with the highest values in vector s. We showed in previous work (Steinberger and Jezek,

2004) that this modification results in a significant improvement over Gong and Liu's method.

3 Using Anaphora Resolution for Summarization

3.1 The case for anaphora resolution

Words are the most basic type of 'term' that can be used to characterize the content of a document. However, being able to identify the most important *objects* mentioned in the document clearly would lead to an improved analysis of what is important in a text, as shown by the following news article cited by Boguraev and Kennedy (1999):

(3) PRIEST IS CHARGED WITH POPE ATTACK

A Spanish priest was charged here today with attempting to murder the Pope. *Juan Fernandez Krohn*, aged 32, was arrested after a man armed with a bayonet approached the Pope while he was saying prayers at Fatima on Wednesday night. According to the police, *Fernandez* told the investigators today that *he* trained for the past six months for the assault. ... If found guilty, *the Spaniard* faces a prison sentence of 15-20 years.

As Boguraev and Kennedy point out, the title of the article is an excellent summary of the content: an entity (the priest) did something to another entity (the pope). Intuitively, understanding that Fernandez and the pope are the central characters is crucial to provide a good summary of texts like these.[2] Among the clues that help us to identify such 'main characters', the fact that an entity is repeatedly mentioned is clearly important.

Purely lexical methods, including the LSA-based methods discussed in the previous section, can only capture part of the information about which entities are frequently repeated in the text. As example (3) shows, stylistic conventions forbid verbatim repetition, hence the six mentions of Fernandez in the text above contain only one lexical repetition, 'Fernandez'. The main problem are pronouns, that tend to share the least lexical similarity with the form used to express the antecedent (and anyway are usually removed by stopword lists, therefore do not get included in the SVD matrix). The form of definite descriptions (*the Spaniard*) doesn't always overlap with that of their antecedent, either, especially when the antecedent was expressed with a proper name. The form of mention which more often overlaps to a degree with previous mentions is proper nouns, and even then at least some way of dealing with acronyms is necessary (cfr. *European Union* / *E.U.*). The motivation for anaphora resolution is that it should tell us which entities are repeatedly mentioned.

[2]It should be noted that for many newspaper articles, indeed many non-educational texts, only a 'entity-centered' structure can be clearly identified, as opposed to a 'relation-centered' structure of the type hypothesized in Rhetorical Structures Theory (Knott et al., 2001; Poesio et al., 2004).

In this work, we tested a mixed approach to integrate anaphoric and word information: using the output of the anaphoric resolver GUITAR to modify the SVD matrix used to determine the sentences to extract. In the rest of this section we first briefly introduce GUITAR, then discuss the two methods we tested to use its output to help summarization.

3.2 GUITAR: A General-Purpose Anaphoric Resolver

The system we used in these experiments, GUITAR (Poesio and Kabadjov, 2004), is an anaphora resolution system designed to be high precision, modular, and usable as an off-the-shelf component of a NL processing pipeline. The current version of the system includes an implementation of the MARS pronoun resolution algorithm (Mitkov, 1998) and a partial implementation of the algorithm for resolving definite descriptions proposed by Vieira and Poesio (2000). The current version of GUITAR does not include methods for resolving proper nouns.

3.2.1 Pronoun Resolution

Mitkov (1998) developed a robust approach to pronoun resolution which only requires input text to be part-of-speech tagged and noun phrases to be identified. Mitkov's algorithm operates on the basis of antecedent-tracking preferences (referred to hereafter as "antecedent indicators"). The approach works as follows: the system identifies the noun phrases which precede the anaphor within a distance of 2 sentences, checks them for gender and number agreement with the anaphor, and then applies genre-specific antecedent indicators to the remaining candidates (Mitkov, 1998). The noun phrase with the highest aggregate score is proposed as antecedent.

3.2.2 Definite Description Resolution

The Vieira / Poesio algorithm (Vieira and Poesio, 2000) attempts to classify each definite description as either direct anaphora, discourse-new, or bridging description. The first class includes definite descriptions whose head is identical to that of their antecedent, as in *a house* … *the house*. Discourse-new descriptions are definite descriptions that refer to objects not already mentioned in the text and not related to any such object. Bridging descriptions are all definite descriptions whose resolution depends on knowledge of relations between objects, such as definite descriptions that refer to an object related to an entity already introduced in the discourse by a relation other than identity, as in *the flat* … *the living room*. The Vieira / Poesio algorithm also attempts to identify the antecedents of anaphoric descriptions and the anchors of bridging ones. The current version of GUITAR incorporates an algorithm for resolving direct anaphora derived quite directly from Vieira / Poesio, as well as a statistical version of the methods for detecting discourse new descriptions (Poesio et al., 2005).

3.3 SVD over Lexical and Anaphoric Terms

SVD can be used to identify the ‘implicit topics’ or main terms of a document not only when on the basis of words, but also of coreference chains, or a mixture of both. We tested two ways of combining these two types of information.

3.3.1 The Substitution Method

The simplest way of integrating anaphoric information with the methods used in our earlier work is to use anaphora resolution simply as a preprocessing stage of the SVD input matrix creation. Firstly, all anaphoric relations are identified by the anaphoric resolver, and anaphoric chains are identified. Then a second document is produced, in which all anaphoric nominal expressions are replaced by the first element of their anaphoric chain. For example, suppose we have the text in (4).

(4) **S1:** *Australia’s new conservative government* on Wednesday began selling *its tough deficit-slashing budget*, which sparked *violent protests by Aborigines, unions, students and welfare groups* even before *it* was announced.

S2: Two days of *anti-budget street protests* preceded *spending cuts* officially unveiled by *Treasurer Peter Costello*.

S3: ”If *we* don’t do *it* now, *Australia* is going to be in *deficit* and debt into the next century.”

S4: As *the protesters* had feared, *Costello* revealed a cut to *the government’s* Aboriginal welfare commission among *the hundreds of measures implemented to claw back the deficit*.

An ideal resolver would find 8 anaphoric chains:

Chain 1 *Australia - we - Australia*

Chain 2 *its new conservative government (Australia’s new conservative government) - the government*

Chain 3 *its tough deficit-slashing budget (Australia’s tough deficit-slashing budget) - it*

Chain 4 *violent protests by Aborigines, unions, students and welfare groups - anti-budget street protests*

Chain 5 *Aborigines, unions, students and welfare groups - the protesters*

Chain 6 *spending cuts - it - the hundreds of measures implemented to claw back the deficit*

Chain 7 *Treasurer Peter Costello - Costello*

Chain 8 *deficit - the deficit*

By replacing each element of the 8 chains above in the text in (4) with the first element of the chain, we get the text in (5).

(5) **S1:** *Australia’s new conservative government* on Wednesday began selling *Australia’s tough deficit-slashing budget*, which sparked *violent protests by Aborigines, unions, students and welfare groups* even before *Australia’s tough deficit-slashing budget* was announced.

S2: Two days of *violent protests by Aborigines, unions, students and welfare groups* preceded *spending cuts* officially unveiled by *Treasurer Peter Costello*.

S3: ”If *Australia* doesn’t do *spending cuts* now, *Australia* is going to be in *deficit* and debt into the next century.”

S4: As *Aborigines, unions, students and welfare groups* had feared, *Treasurer Peter Costello* revealed a cut to *Australia’s new conservative government’s* Aboriginal welfare commission among *the spending cuts*.

This text is then used to create the SVD input matrix, as done in the first system.

3.3.2 The Addition Method

An alternative approach is to use SVD to identify ‘topics’ on the basis of two types of ’terms’: terms in the lexical sense (i.e., words) and terms in the sense of objects, which can be represented by anaphoric chains. In other words, our representation of sentences would specify not only if they contain a certain word, but also if they contain a mention of a discourse entity (See Figure 1.) This matrix would then be used as input to SVD.

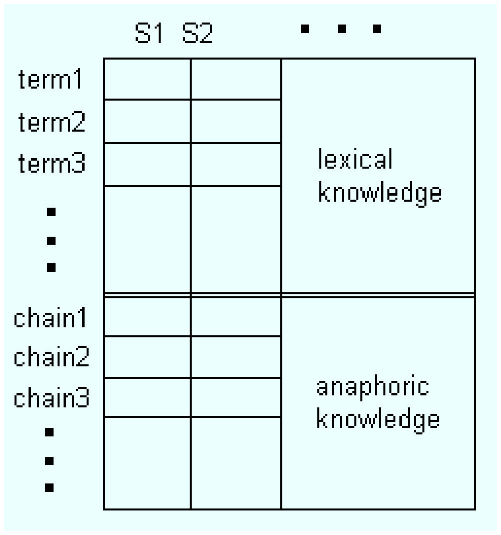


Figure 1: Addition method.

The chain 'terms' tie together sentences that contain the same anaphoric chain. If the terms are lexically the same (direct anaphors - like *deficit* and *the deficit*) the basic summarizer works sufficiently. However, Gong and Liu showed that the best weighting scheme is boolean (i.e., all terms have the same weight); our own previous results confirmed this. The advantage of the addition method is the opportunity to give higher weights to anaphors.

4 Evaluation

4.1 The CAST Corpus

To evaluate our system, we used the corpus of manually produced summaries created by the CAST project[3] (Orasan et al., 2003). The CAST corpus contains news articles taken from the Reuters Corpus and a few popular science texts from the British National Corpus. It contains information about the importance of the sentences (Hasler et al., 2003). Sentences are marked as **essential** or **important**. The corpus also contains annotations for linked sentences, which are not significant enough to be marked as important/essential, but which have to be considered as they contain information essential for the understanding of the content of other sentences marked as essential/important.

[3]The goal of this project was to investigate to what extent Computer-Aided Summarization can help humans to produce high quality summaries with less effort.

Four annotators were used for the annotation, three graduate students and one postgraduate. Three of the annotators were native English speakers, and the fourth had advanced knowledge of English. Unfortunately, not all of the documents were annotated by all of the annotators. To maximize the reliability of the summaries used for evaluation, we chose the documents annotated by the greatest number of the annotators; in total, our evaluation corpus contained 37 documents.

For acquiring manual summaries at specified lengths and getting the sentence scores (for relative utility evaluation) we assigned a score 3 to the sentences marked as essential, a score 2 to important sentences and a score 1 to linked sentences. The sentences with highest scores are then selected for ideal summary (at specified lenght).

4.2 Evaluation Measures

Evaluating summarization is a notoriously hard problem, for which standard measures like Precision and Recall are not very appropriate. The main problem with P&R is that human judges often disagree what are the top n% most important sentences in a document. Using P&R creates the possibility that two equally good extracts are judged very differently. Suppose that a manual summary contains sentences [1 2] from a document. Suppose also that two systems, A and B, produce summaries consisting of sentences [1 2] and [1 3], respectively. Using P&R, system A will be ranked much higher than system B. It is quite possible that sentences 2 and 3 are equally important, in which case the two systems should get the same score.

To address the problem with precision and recall we used a combination of evaluation measures. The first of these, relative utility (RU) (Radev et al., 2000) allows model summaries to consist of sentences with variable ranking. With RU, the model summary represents all sentences of the input document with confidence values for their inclusion in the summary. For example, a document with five sentences [1 2 3 4 5] is represented as [1/5 2/4 3/4

Evaluation Method	Lexical LSA	Manual Substitution	Manual Addition
Relative Utility	0.595	0.573	**0.662**
F-score	0.420	0.410	**0.489**
Cosine Similarity	0.774	0.806	**0.823**
Main Topic Similarity	0.686	0.682	**0.747**

Table 1: Evaluation of the manual annotation improvement - summarization ratio: 15%.

Evaluation Method	Lexical LSA	Manual Substitution	Manual Addition
Relative Utility	0.645	0.662	**0.688**
F-score	0.557	0.549	**0.583**
Cosine Similarity	0.863	0.878	**0.886**
Main Topic Similarity	0.836	0.829	**0.866**

Table 2: Evaluation of the manual annotation improvement - summarization ratio: 30%.

4/1 5/2]. The second number in each pair indicates the degree to which the given sentence should be part of the summary according to a human judge. This number is called the utility of the sentence. Utility depends on the input document, the summary length, and the judge. In the example, the system that selects sentences [1 2] will not get a higher score than a system that chooses sentences [1 3] given that both summaries [1 2] and [1 3] carry the same number of utility points (5+4). Given that no other combination of two sentences carries a higher utility, both systems [1 2] and [1 3] produce optimal extracts. To compute relative utility, a number of judges, $(N \geq 1)$ are asked to assign utility scores to all n sentences in a document. The top e sentences according to utility score[4] are then called a sentence extract of size e. We can then define the following system performance metric:

$$(6) \qquad RU = \frac{\sum_{j=1}^{n} \delta_j \sum_{i=1}^{N} u_{ij}}{\sum_{j=1}^{n} \epsilon_j \sum_{i=1}^{N} u_{ij}},$$

where u_{ij} is a utility score of sentence j from annotator i, ϵ_j is 1 for the top e sentences according to the sum of utility scores from all judges and δ_j is equal to 1 for the top e sentences extracted by the system. For details see (Radev et al., 2000).

The second measure we used is Cosine Similarity, according to the standard formula:

$$(7) \qquad cos(X,Y) = \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i (x_i)^2} \cdot \sqrt{\sum_i (y_i)^2}},$$

where X and Y are representations of a system summary and its reference summary based on the vector space model. The third measure is Main Topic Similarity. This is a content-based evaluation method based on measuring the cosine of the angle between first left singular vectors of a system summary's and its reference summary's SVDs. (For details see (Steinberger and Jezek, 2004).) Finally, we measured ROUGE scores, with the same settings as in the Document Understanding Conference (DUC) 2004.

4.3 How Much May Anaphora Resolution Help? An Upper Bound

We annotated all the anaphoric relations in the 37 documents in our evaluation corpus by hand using the annotation tool MMAX (Mueller and Strube, 2003).[5] Apart from measuring the performance of GUITAR over the corpus, this allowed us to establish the upper bound on the performance improvements that could be obtained by adding an anaphoric resolver to our summarizer. We tested both methods of adding the anaphoric knowledge to the summarizer discussed above. Results for the 15% and 30% ratios[6] are presented in Tables 1 and 2. The baseline is our own previously developed LSA-based summarizer without anaphoric knowledge. The result is that the substitution method did not lead to significant improvement, but the addition method did:

[4]In the case of ties, some arbitrary but consistent mechanism is used to decide which sentences should be included in the summary.

[5]We annotated personal pronouns, possessive pronouns, definite descriptions and also proper nouns, who will be handled by a future GUITAR version.

[6]We used the same summarization ratios as in CAST.

Evaluation Method	Lexical LSA	CAST	GUITAR Substitution	GUITAR Addition
Relative Utility	0.595	0.527	0.530	**0.640**
F-score	0.420	0.348	0.347	**0.441**
Cosine Similarity	0.774	0.726	0.804	**0.805**
Main Topic Similarity	0.686	0.630	0.643	**0.699**

Table 3: Evaluation of the GUITAR improvement - summarization ratio: 15%.

Evaluation Method	Lexical LSA	CAST	GUITAR Substitution	GUITAR Addittion
Relative Utility	0.645	0.618	0.626	**0.678**
F-score	0.557	0.522	0.524	**0.573**
Cosine Similarity	0.863	0.855	0.873	**0.879**
Main Topic Similarity	0.836	0.810	0.818	**0.868**

Table 4: Evaluation of the GUITAR improvement - summarization ratio: 30%.

addition could lead to an improvement in Relative Utility score from .595 to .662 for the 15% ratio, and from .645 to .688 for the 30% ratio. Both of these improvements were significant by t-test at 95% confidence.

4.4 Results with GUITAR

To use GUITAR, we first parsed the texts using Charniak's parser (Charniak, 2000). The output of the parser was then converted into the MAS-XML format expected by GUITAR by one of the preprocessors that come with the system. (This step includes heuristic methods for guessing agreement features.) Finally, GUITAR was ran to add anaphoric information to the files. The resulting files were then processed by the summarizer.

GUITAR achieved a precision of 56% and a recall of 51% over the 37 documents. For definite description resolution, we found a precision of 69% and a recall of 53%; for possessive pronoun resolution, the precision was 53%, recall was 53%; for personal pronouns, the precision was 44%, recall was 46%.

The results with the summarizer are presented in Tables 3 and 4 (relative utility, f-score, cosine, and main topic). The contribution of the different anaphora resolution components is addressed in (Kabadjov et al., 2005). All versions of our summarizer (the baseline version without anaphora resolution and those using substitution and addition) outperformed the CAST summarizer, but we have to emphasize that CAST did not aim at producing a high-performance generic summarizer; only a system that could be easily used for didactical purposes. However, our tables also show that using GUITAR and the addition method lead to significant improvements over our baseline LSA summarizer. The improvement in Relative Utility measure was significant by t-test at 95% confidence. Using the ROUGE measure we obtained improvement (but not significant). On the other hand, the substitution method did not lead to significant improvements, as was to be expected given that no improvement was obtained with 'perfect' anaphora resolution (see previous section).

5 Conclusion and Further Research

Our main result in this paper is to show that using anaphora resolution in summarization can lead to significant improvements, not only when 'perfect' anaphora information is available, but also when an automatic resolver is used, provided that the anaphoric resolver has reasonable performance. As far as we are aware, this is the first time that such a result has been obtained using standard evaluation measures over a reference corpus. We also showed however that the way in which anaphoric information is used matters: with our set of documents at least, substitution would not result in significant improvements even with perfect anaphoric knowledge.

Further work will include, in addition to extending the set of documents and testing the system with other collections, evaluating the improvement to be achieved by adding a proper noun resolution algorithm to GUITAR.

References

S. Azzam, K. Humphreys and R. Gaizauskas. 1999. Using coreference chains for text summarization. In *Proceedings of the ACL Workshop on Coreference*. Maryland.

B. Baldwin and T. S. Morton. 1998. Dynamic coreference-based summarization. In *Proceedings of EMNLP*. Granada, Spain.

R. Barzilay and M. Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the ACL/EACL Workshop on Intelligent Scalable Text Summarization*. Madrid, Spain.

S. Bergler, R. Witte, M. Khalife, Z. Li, and F. Rudzicz. 2003. Using Knowledge-poor Coreference Resolution for Text Summarization. In *Proceedings of DUC*. Edmonton.

M. W. Berry, S. T. Dumais and G. W. O'Brien. 1995. Using Linear Algebra for Intelligent IR. In *SIAM Review*, 37(4).

B. Boguraev and C. Kennedy. 1999. Salience-based content characterization of text documents. In I. Mani and M. T. Maybury (eds), *Advances in Automatic Text Summarization,* MIT Press. Cambridge, MA.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*. Philadelphia.

F. Y. Y. Choi, P. Wiemer-Hastings and J. D. Moore. 2001. Latent Semantic Analysis for Text Segmentation. In *Proceedings of EMNLP*. Pittsburgh.

Y. Gong and X. Liu. 2002. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In *Proceedings of ACM SIGIR*. New Orleans.

L. Hasler, C. Orasan and R. Mitkov. 2003. Building better corpora for summarization. In *Proceedings of Corpus Linguistics*. Lancaster, United Kingdom.

E. Hovy and C. Lin. 1997. Automated text summarization in SUMMARIST. In *ACL/EACL Workshop on Intelligent Scalable Text Summarization*. Madrid, Spain.

M. A. Kabadjov, M. Poesio and J. Steinberger. 2005. Task-Based Evaluation of Anaphora Resolution: The Case of Summarization. In *RANLP Workshop "Crossing Barriers in Text Summarization Research"*. Borovets, Bulgaria.

A. Knott, J. Oberlander, M. O'Donnell, and C. Mellish. 2001. Beyond elaboration: The interaction of relations and focus in coherent text. In Sanders, T., Schilperoord, J., and Spooren, W. (eds), *Text representation: linguistic and psycholinguistic aspects*. John Benjamins.

T. K. Landauer and S. T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. In *Psychological Review*, 104, 211-240.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proceedings of COLING*. Montreal.

C. Mueller and M. Strube. 2001. MMAX: A Tool for the Annotation of Multi-modal Corpora. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. Seattle.

C. Orasan, R. Mitkov and L. Hasler. 2003. CAST: a Computer-Aided Summarization Tool. In *Proceedings of EACL*. Budapest, Hungary.

M. Poesio and M. A. Kabadjov. 2004. A General-Purpose, off-the-shelf Anaphora Resolution Module: Implementation and Preliminary Evaluation. In *Proceedings of LREC*. Lisbon, Portugal.

M. Poesio, R. Stevenson, B. Di Eugenio, and J. M. Hitzeman. 2004. Centering: A parametric theory and its instantiations. *Computational Linguistics*, 30(3).

M. Poesio, M. A. Kabadjov, R. Vieira, R. Goulart, and O. Uryupina. 2005. Do discourse-new detectors help definite description resolution? In *Proceedings of IWCS*. Tilburg, The Netherlands.

D. R. Radev, H. Jing, and M. Budzikowska. 2000. Centroid-based summarization of multiple documents. In *ANLP/NAACL Workshop on Automatic Summarization*. Seattle.

J. Steinberger and K. Jezek. 2004. Text Summarization and Singular Value Decomposition. In *Proceedings of ADVIS*. Izmir, Turkey.

R. Stuckardt. 2003. Coreference-Based Summarization and Question Answering: a Case for High Precision Anaphor Resolution. In *International Symposium on Reference Resolution.* Venice, Italy.

R. Vieira and M. Poesio. 2000. An empirically-based system for processing definite descriptions. In *Computational Linguistics*, 26(4).

Data-driven Approaches for Information Structure Identification

Oana Postolache,
Ivana Kruijff-Korbayová
University of Saarland,
Saarbrücken, Germany
{oana,korbay}@coli.uni-saarland.de

Geert-Jan M. Kruijff
German Research Center for
Artificial Intelligence (DFKI GmbH)
Saarbrücken, Germany
gj@dfki.de

Abstract

This paper investigates automatic identification of Information Structure (IS) in texts. The experiments use the Prague Dependency Treebank which is annotated with IS following the Praguian approach of Topic Focus Articulation. We automatically detect t(opic) and f(ocus), using node attributes from the treebank as basic features and derived features inspired by the annotation guidelines. We present the performance of decision trees (C4.5), maximum entropy, and rule induction (RIPPER) classifiers on all tectogrammatical nodes. We compare the results against a baseline system that always assigns f(ocus) and against a rule-based system. The best system achieves an accuracy of 90.69%, which is a 44.73% improvement over the baseline (62.66%).

1 Introduction

Information Structure (IS) is a partitioning of the content of a sentence according to its relation to the discourse context. There are numerous theoretical approaches describing IS and its semantics (Halliday, 1967; Sgall, 1967; Vallduví, 1990; Steedman, 2000) and the terminology used is diverse — see (Kruijff-Korbayová and Steedman, 2003) for an overview. However, all theories consider at least one of the following two distinctions: (i) a Topic/Focus[1] distinction that divides the linguistic meaning of the sentence into parts that link the sentence content to the discourse context, and other parts that advance the discourse, i.e., add or modify information; and (ii) a background/kontrast[2] distinction between parts of the utterance which contribute to distinguishing its actual content from alternatives the context makes available.

Information Structure is an important factor in determining the felicity of a sentence in a given context. Applications in which IS is crucial are text-to-speech systems, where IS helps to improve the quality of the speech output (Prevost and Steedman, 1994; Kruijff-Korbayová et al., 2003; Moore et al., 2004), and machine translation, where IS improves target word order, especially that of free word order languages (Stys and Zemke, 1995).

Existing theories, however, state their principles using carefully selected illustrative examples. Because of this, they fail to adequately explain how different linguistic dimensions cooperate to realize Information Structure.

In this paper we describe data-driven, machine learning approaches for automatic identification of Information Structure; we describe what aspects of IS we deal with and report results of the performance of our systems and make an error analysis. For our experiments, we use the Prague Dependency Treebank (PDT) (Hajič, 1998). PDT follows the theory of Topic-Focus Articulation (Hajičová et al., 1998) and to date is the only corpus annotated with IS. Each node of the underlying structure of sentences in PDT is annotated with a TFA value: t(opic), differentiated in contrastive and non-contrastive, and f(ocus). Our system identifies these two TFA values automatically. We trained three different clas-

[1] We use the Praguian terminology for this distinction.

[2] The notion 'kontrast' with a 'k' has been introduced in (Vallduví and Vilkuna, 1998) to replace what Steedman calls 'focus', and to avoid confusion with other definitions of focus.

sifiers, C4.5, RIPPER and MaxEnt using basic features from the treebank and derived features inspired by the annotation guidelines. We evaluated the performance of the classifiers against a baseline system that simulates the preprocessing procedure that preceded the manual annotation of PDT, by always assigning f(ocus), and against a rule-based system which we implemented following the annotation instructions. Our best system achieves a 90.69% accuracy, which is a 44.73% improvement over the baseline (62.66%).

The organization of the paper is as follows. Section 2 describes the Prague Dependency Treebank and the Praguian approach of Topic-Focus Articulation, from two perspectives: of the theoretical definition and of the annotation guidelines that have been followed to annotate the PDT. Section 3 presents our experiments, the data settings, results and error analysis. The paper closes with conclusions and issues for future research (Section 4).

2 Prague Dependency Treebank

The Prague Dependency Treebank (PDT) consists of newspaper articles from the Czech National Corpus (Čermák, 1997) and includes three layers of annotation:

1. The morphological layer gives a full morphemic analysis in which 13 categories are marked for all sentence tokens (including punctuation marks).
2. The analytical layer, on which the "surface" syntax (Hajič, 1998) is annotated, contains analytical tree structures, in which every token from the surface shape of the sentence has a corresponding node labeled with main syntactic functions like SUBJ, PRED, OBJ, ADV.
3. The tectogrammatical layer renders the deep (underlying) structure of the sentence (Sgall et al., 1986; Hajičová et al., 1998). Tectogrammatical tree structures (TGTSs) contain nodes corresponding only to the autosemantic words of the sentence (e.g., no preposition nodes) and to deletions on the surface level; the condition of projectivity is obeyed, i.e., no crossing edges are allowed; each node of the tree is assigned a functor such as ACTOR, PATIENT, ADDRESSEE, ORIGIN, EFFECT, the repertoire of which is very rich; elementary coreference links are annotated for pronouns.

2.1 Topic-Focus Articulation (TFA)

The tectogrammatical level of the PDT was motivated by the ever increasing need for large corpora to include not only morphological and syntactic information but also semantic and discourse-related phenomena. Thus, the tectogrammatical trees have been enriched with features indicating the information structure of sentences which is a means of showing their contextual potential.

In the Praguian approach to IS, the content of the sentence is divided into two parts: the Topic is "what the sentence is about" and the Focus represents the information asserted about the Topic. A prototypical declarative sentence asserts that its Focus holds (or does not hold) about its Topic: Focus(Topic) or not-Focus(Topic).

The TFA definition uses the distinction between Context-Bound (CB) and Non-Bound (NB) parts of the sentence. To distinguish which items are CB and which are NB, the question test is applied, (i.e., the question for which a given sentence is the appropriate answer is considered). In this framework, weak and zero pronouns and those items in the answer which reproduce expressions present in the question (or associated to those present) are CB. Other items are NB.

In example (1), (b) is the sentence under investigation, in which CB and NB items are marked. Sentence (a) is the context in which the sentence (b) is uttered, and sentence (c) is the question for which the sentence (b) is an appropriate answer:

(1) (a) Tom and Mary both came to John's party.
(b) John$_{CB}$ invited$_{CB}$ only$_{NB}$ her$_{NB}$.
(c) Whom did John invite?

It should be noted that the CB/NB distinction is not equivalent to the given/new distinction, as the pronoun "her" is NB although the cognitive entity, Mary, has already been mentioned in the discourse (therefore is given).

The following rules determine which lexical items (CB or NB) belong to the Topic or to the Focus of the sentence (Hajičová et al., 1998; Hajičová and Sgall, 2001):

1. The main verb and any of its direct dependents belong to the Focus if they are NB;
2. Every item that does not depend directly on the main verb and is subordinated to a Focus element belongs to the Focus (where "subordinated to" is defined as the irreflexive transitive closure of "depend on");
3. If the main verb and all its dependents are CB, then those dependents d_i of the verb which have subordinated items s_m that are NB are called 'proxi foci'; the items s_m together with all items subordinated to them belong to the Focus ($i, m > 1$);
4. Every item not belonging to the Focus according to 1 – 3 belongs to the Topic.

Applying these rules for the sentence (b) in example (1) we find the Topic and the Focus of the sentence: [John invited]$_{Topic}$ [only her]$_{Focus}$.

It is worth mentioning that although most of the time, CB items belong to the Topic and NB items belong to the Focus (as it happens in our example too), there may be cases when the Focus contains some NB items and/or the Topic contains some CB items. Figure 1 shows such configurations: in the top-left corner the tectogrammatical representation of sentence (1) (b) is presented together with its Topic-Focus partitioning. The other three configurations are other possible tectogrammatical trees with their Topic-Focus partitionings; the top-right one corresponds to the example (2), the bottom-left to (3), and bottom-right to (4).

(2) Q: Which teacher did Tom meet?
A: Tom$_{CB}$ met$_{CB}$ the teacher$_{CB}$ of chemistry$_{NB}$.

(3) Q: What did he think about the teachers?
A: He$_{CB}$ liked$_{NB}$ the teacher$_{CB}$ of chemistry$_{NB}$.

(4) Q: What did the teachers do?
A: The teacher$_{CB}$ of chemistry$_{NB}$ met$_{NB}$ his$_{CB}$ pupils$_{NB}$.

2.2 TFA annotation

Within PDT, the TFA attribute has been annotated for all nodes (including the restored ones) from the tectogrammatical level. Instructions for the assignment of the TFA attribute have been specified in

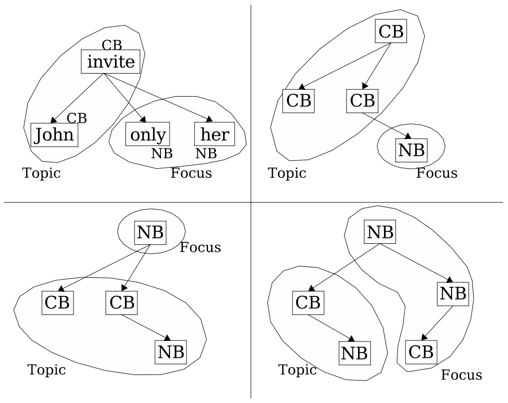


Figure 1: Topic-Focus partitionings of tectogrammatical trees.

(Buráňová et al., 2000) and are summarized in Table 1. These instructions are based on the surface word order, the position of the sentence stress (intonation center – IC)[3] and the canonical order of the dependents.

The TFA attribute has three values:

1. t — for non-contrastive CB items;
2. f — for NB items;
3. c — for contrastive CB items.

In this paper, we do not distinguish between contrastive and non-contrastive items, considering both of them as being just t. In the PDT annotation, the notation t (from topic) and f (from focus) was chosen to be used because, as we mentioned earlier, in the most common cases and in prototypical sentences, t-items belong to the Topic and f-items to the Focus.

Prior the manual annotation, the PDT corpus was preprocessed to mark all nodes with the TFA attribute of f, as it is the most common value. Then the annotators corrected the value according to the guidelines in Table 1.

Figure 2 illustrates the tectogramatical tree structure of the following sentence:

(5)	Sebevědomím	votroků	to	ale	neotřáslo.
	self-confidence	bastards	it	but not	shake

'But it did not shake the self-confidence of those bastards'.

[3] In the PDT the intonation center is not annotated. However, the annotators were instructed to use their judgement where the IC would be if they uttered the sentence.

1.	The bearer of the IC (typically, the rightmost child of the verb)	f
2.	If IC is not on the rightmost child, everything after IC	t
3.	A left-side child of the verb (unless it carries IC)	t
4.	The verb and the right children of the verb before the f-node (cf. 1) that are canonically ordered	f
5.	Embedded attributes (unless repeated or restored)	f
6.	Restored nodes	t
7.	Indexical expressions (*já* I, *ty* you, *těd* now, *tady* here), weak pronouns, pronominal expressions with a general meaning (*někdo* somebody, *jednou* once) (unless they carry IC)	t
8.	Strong forms of pronouns not preceded by a preposition (unless they carry IC)	t

Table 1: Annotation guidelines; IC = Intonation Center.

Each node is labeled with the corresponding word's lemma, the TFA attribute, and the functor attribute. For example, *votroků* has lemma *votrok*, the TFA attribute *f*, and the functor *APP* (appurtenance).

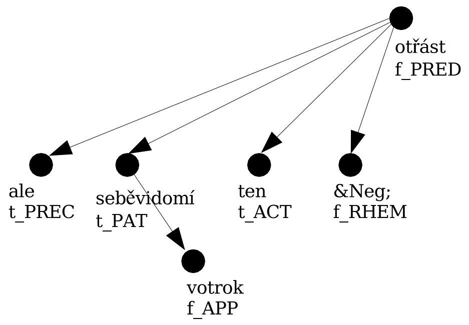


Figure 2: Tectogramatical tree annotated with t/f.

In order to measure the consistency of the annotation, Interannotator Agreement has been measured (Veselá et al., 2004).[4] During the annotation process, there were four phases in which parallel annotations have been performed; a sample of data was chosen and annotated in parallel by three annotators.

Agreement	1	2	3	4	Avg
t/c/f	81.32	81.89	76.21	89.57	82.24
t/f	85.42	83.94	84.18	92.15	86.42

Table 2: Interannotator Agreement for TFA assignment in PDT 2.0.

The agreement for each of the four phases, as well as an average agreement, is shown in Table 2. The second row of the table displays the percentage of nodes for which all three annotators assigned the same TFA value (be it t, c or f). Because in our experiments we do not differentiate between t and c, considering both as t, we computed, in the last row of the table, the agreement between the three annotators after replacing the TFA value c with t.[5]

3 Identification of topic and focus

In this section we present data-driven, machine learning approaches for automatic identification of Information Structure. For each tectogrammatical node we detect the TFA value t(opic) or f(ocus) (that is CB or NB). With these values one can apply the rules presented in Subsection 2.1 in order to find the Topic-Focus partitioning of each sentence.

3.1 Experimental settings

Our experiments use the tectogrammatical trees from The Prague Dependency Treebank 2.0.[6] Statistics of the experimental data are shown in Table 3.

Our goal is to automatically label the tectogrammatical nodes with topic or focus. We built machine learning models based on three different well known techniques, decision trees (C4.5), rule induction (RIPPER) and maximum entropy (MaxEnt), in order to find out which approach is the most suitable for our task. For C4.5 and RIPPER we use the Weka implementations (Witten and Frank, 2000) and for MaxEnt we use the openNLP package.[7]

[4] In their paper the authors don't give Kappa values, nor the complete information needed to compute a Kappa statistics ourselves.

[5] In (Veselá et al., 2004), the number of cases when the annotators disagreed when labeling t or c is reported; this allowed us to compute the t/f agreement, by disregarding this number.

[6] We are grateful to the researchers at the Charles University in Prague for providing us the data before the PDT 2.0 official release.

[7] `http://maxent.sourceforge.net/`

PDT DATA	TRAIN	DEV	EVAL	TOTAL
#files	2,536 80%	316 10%	316 10%	3,168 100%
#sentences	38,737 78.3%	5,228 10.6%	5,477 11.1%	49,442 100%
#tokens	652,700 78.3%	87,988 10.6%	92,669 11.1%	833,356 100%
#tecto-nodes	494,759 78.3%	66,711 10.5%	70,323 11.2%	631,793 100%

Table 3: PDT data: Statistics for the training, development and evaluation sets.

All our models use the same set of 35 features (presented in detail in Appendix A), divided in two types:

1. Basic features, consisting of attributes of the tectogrammatical nodes whose values were taken directly from the treebank annotation. We used a total of 25 basic features, that may have between 2 and 61 values.

2. Derived features, inspired by the annotation guidelines. The derived features are computed using the dependency information from the tectogrammatical level of the treebank and the surface order of the words corresponding to the nodes.[8] We also used lists of forms of Czech pronouns that are used as weak pronouns, indexical expressions, pronouns with general meaning, or strong pronouns. All the derived features have boolean values.

3.2 Results

The classifiers were trained on 494,759 instances (78.3%) (cf. Table 3) (tectogrammatical nodes) from the training set. The performance of the classifiers was evaluated on 70,323 instances (11.2%) from the evaluation set. We compared our models against a baseline system that assigns focus to all nodes (as it is the most common value) and against a deterministic, rule-based system, that implements the instructions from the annotation guidelines.

Table 4 shows the percentages of correctly classified instances for our models. We also performed a 10-fold cross validation, which for C4.5 gives accuracy of 90.62%.

BASELINE	RULE-BASED	C4.5	RIPPER	MAXENT
62.66	58.92	90.69	88.46*	88.97

Table 4: Correctly classified instances (the numbers are given as percentages). *The RIPPER classifier was trained with only 40% of the training data.

The baseline value is considerably high due to the topic/focus distribution in the test set (a similar distribution characterizes the training set as well). The rule-based system performs very poorly, although it follows the guidelines according to which the data was annotated. This anomaly is due to the fact that the intonation center of the sentence, which plays a very important role in the annotation, is not marked in the corpus, thus the rule-based system doesn't have access to this information.

The results show that all three models perform much better than the baseline and the rule-based system. We used the χ^2 test to examine if the difference between the three classifiers is statistically significant. The C4.5 model significantly outperforms the MaxEnt model (χ^2 = 113.9, $p < 0.001$) and the MaxEnt model significantly outperforms the RIPPER model although with a lower level of confidence (χ^2 = 9.1, $p < 0.01$).

The top of the decision tree generated by C4.5 in the training phase looks like this:

```
coref = true
|    is_member = true
|    |    POS = ...
|    is_member = false
|    |    is_rightmost = ...
coref = false
|    is_generated = true
|    |    nodetype = ...
|    is_generated = false
|    |    iterativeness = ...
```

It is worth mentioning that the RIPPER classifier was built with only 40% of the training set (with more data, the system crashes due to insufficient memory). Interestingly and quite surprisingly, the values of all three classifiers are actually greater than the interannotator agreement which has an average of 86.42%.

What is the cause of the classifiers' success? How come that they perform better than the annotators themselves? Is it because they take advantage of a

[8] In the tectogramatical level in the PDT, the order of the nodes has been changed during the annotation process of the TFA attribute, so that all t items precede all f items. Our features use the surface order of the words corresponding to the nodes.

large amount of training data? To answer this question we have computed the learning curves. They are shown in the figure 3, which shows that, actually, after using only 1% of the training data (4,947 instances), the classifiers already perform very well, and adding more training data improves the results only slightly. On the other hand, for RIPPER, adding more data causes a decrease in performance, and as we mentioned earlier, even an impossibility of building a classifier.

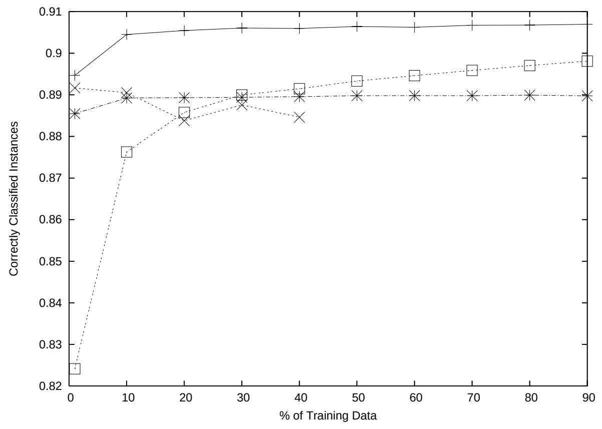


Figure 3: Learning curves for C4.5 (+), RIPPER(×), MaxEnt(∗) and a naïve predictor (□) (introduced in Section 3.3).

3.3 Error Analysis

If errors don't come from the lack of training data, then where do they come from? To answer this question we performed an error analysis. For each instance (tectogrammatical node), we considered its *context* as being the set of values for the features presented in Appendix A. Table 5 displays in the second column the number of all contexts. The last three columns divide the contexts in three groups:

1. Only t — all instances having these contexts are assigned t;
2. Only f — all instances having these contexts are assigned f;
3. Ambiguous — some instances that have these contexts are assigned t and some other are assigned f.

The last row of the table shows the number of instances for each type of context, in the training data.

	All	Only t	Only f	Ambiguous
#contexts	27,901	9,901	13,009	4,991
#instances	494,759 100%	94,056 19.01%	42,048 8.49%	358,655 72.49%

Table 5: Contexts & Instances in the training set.

Table 5 shows that the source of ambiguity (and therefore of errors) stays in 4,991 contexts that correspond to nodes that have been assigned both t and f. Moreover these contexts yield the largest amount of instances (72.49%). We investigated further these ambiguous contexts and we counted how many of them correspond to a set of nodes that are mostly assigned t (#t $>$ #f), respectively f (#t $<$ #f), and how many are highly ambiguous (half of the corresponding instances are assigned t and the other half f (#t $=$ #f)). The numbers, shown in Table 6, suggest that in the training data there are 41,851 instances (8.45%) (the sum of highlighted numbers in the third row of the Table 6) that are exceptions, meaning they have contexts that usually correspond to instances that are assigned the other TFA value. There are two explanations for these exceptions: either they are part of the annotators disagreement, or they have some characteristics that our set of features fail to capture.

	#t $>$ #f	#t $=$ #f	#t $<$ #f
#ambiguous contexts	998	833	3,155

#instances	t=50,722
	f=**4,854**
	all=55,576
	11.23%
		t=**602**
		f=**602**
		all=1,204
		0.24%
			t=**35,793**
			f=266,082
			all=301,875
			61.01%

Table 6: Ambiguous contexts in the training data.

The error analysis led us to the idea of implementing a naïve predictor. This predictor trains on the training set, and divides the contexts into five groups. Table 7 describes these five types of contexts and displays the TFA value assigned by the naïve predictor for each type.

If an instance has a context of type #t $=$ #f, we decide to assign f because this is the most common value. Also, for the same reason, new contexts in the test set that don't appear in the training set are assigned f.

The performance of the naïve predictor on the evaluation set is 89.88% (correctly classified instances), a value which is significantly higher than

Context Type	In the training set, instances with a context of this type are:	Predicted TFA value
Only t	all t	t
Only f	all f	f
#t > #f	more t than f	t
#t = #f	half t, half f	f
#t < #f	more f than t	f
unseen	not seen	f

Table 7: Naïve Predictor: its TFA prediction for each type of context.

the one obtained by the MaxEnt and RIPPER classifiers (χ^2 = 30.7, $p < 0.001$ and respectively χ^2 = 73.3, $p < 0.001$), and comparable with the C4.5 value, although the C4.5 classifier still performs significantly better (χ^2 = 26.3, $p < 0.001$).

To find out whether the naïve predictor would improve if we added more data, we computed the learning curve, shown in Figure 3. Although the curve is slightly more abrupt than the ones of the other classifiers, we do not have enough evidence to believe that more data in the training set would bring a significant improvement. We calculated the number of new contexts in the development set, and although the number is high (2,043 contexts), they correspond to only 2,125 instances. This suggests that the new contexts that may appear are very rare, therefore they cannot yield a big improvement.

4 Conclusions

In this paper we investigated the problem of learning Information Structure from annotated data. The contribution of this research is to show for the first time that IS can be successfuly recovered using mostly syntactic features. We used the Prague Dependency Treebank which is annotated with Information Structure following the Praguian theory of Topic Focus Articulation. The results show that we can reliably identify t(opic) and f(ocus) with over 90% accuracy while the baseline is at 62%.

Issues for further research include, on the one hand, a deeper investigation of the Topic-Focus Articulation in the Prague Dependency Treebank of Czech, by improving the feature set, considering also the distinction between contrastive and non-contrastive t items and, most importantly, by investigating how we can use the t/f annotation in PDT (and respectively our results) in order to detect the Topic/Focus partitioning of the whole sentence.

We also want to benefit from our experience with the Czech data in order to create an English corpus annotated with Information Structure. We have already started to exploit a parallel English-Czech corpus, in order to transfer to the English version the topic/focus labels identified by our systems.

References

Eva Buráňová, Eva Hajičová, and Petr Sgall. 2000. Tagging of very large corpora: Topic-Focus Articulation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 139–144.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of valency and Meaning. Studies in Honor of Jarmila Panevová*. Karolinum, Prague.

Eva Hajičová and Petr Sgall. 2001. Topic-focus and salience. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, pages 268–273, Toulose, France.

Eva Hajičová, Barbara Partee, and Petr Sgall. 1998. Topic-focus articulation, tripartite structures, and semantic content. In *Studies in Linguistics and Philosophy*, number 71. Dordrecht: Kluwer.

M. Halliday. 1967. Notes on transitivity and theme in english, part ii. *Journal of Linguistic*, (3):199–244.

Ivana Kruijff-Korbayová and Mark Steedman. 2003. Discourse and Information Structure. *Journal of Logic, Language and Information*, (12):249–259.

Ivana Kruijff-Korbayová, Stina Erricson, Kepa J. Rodrígues, and Elena Karagjosova. 2003. Producing Contextually Appropriate Intonation in an Information-State Based Dialog System. In *Proceeding of European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. 2004. Generating Tailored, Comparative Description in Spoken Dialogue. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Sociey Conference*.

Scott Prevost and Mark Steedman. 1994. Information Based Intonation Synthesis. In *Proceedings of the ARPA Workshop on Human Language Technology*, Princeton, USA.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects.* Reidel, Dordrecht.

Petr Sgall. 1967. Functional sentence perspective in a generative description. *Prague Studies in Mathematical Linguistics*, (2):203–225.

Mark Steedman. 2000. Information Structure and the syntax-phonology interface. *Linguistic Inquiry*, (34):649–689.

Malgorzata Stys and Stefan Zemke. 1995. Incorporating Discourse Aspects in English-Polish MT: Towards Robust Implementation. In *Recent Advances in NLP*, Velingrad, Bulgaria.

Enrich Vallduví and Maria Vilkuna. 1998. On rheme and kontrast. In P. Culicover and L. McNally, editors, *Syntax and Semantics Vol 29: The Limits of Syntax*. Academic Press, San Diego.

Enrich Vallduví. 1990. *The information component.* Ph.D. thesis, University of Pennsylvania.

František Čermák. 1997. Czech National Corpus: A Case in Many Contexts. *International Journal of Corpus Linguistics*, (2):181–197.

Kateřina Veselá, Jiří Havelka, and Eva Hajičova. 2004. Annotators' Agreement: The Case of Topic-Focus Articulation. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2004)*.

Ian H. Witten and Eibe Frank. 2000. *Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, San Francisco.

Appendix A

In this appendix we provide a full list of the feature names and the values they take (a feature for MaxEnt being a combination of the name, value and the prediction).

BASIC FEATURE	POSSIBLE VALUES
nodetype	complex, atom, dphr, list, qcomplex
is_generated	true, false
functor	ACT, LOC, DENOM, APP, PAT, DIR1, MAT, RSTR, THL, TWHEN, REG, CPHR, COMPL, MEANS, ADDR, CRIT, TFHL, BEN, ORIG, DIR3, TTILL, TSIN, MANN, EFF, ID, CAUS, CPR, DPHR, AIM, EXT, ACMP, THO, DIR2, RESTR, TPAR, PAR, COND, CNCS, DIFF, SUBS, AUTH, INTT, VOCAT, TOWH, ATT, RHEM, TFRWH, INTF, RESL, PREC, PRED, PARTL, HER, MOD, CONTRD
coref	true, false
afun	Pred, Pnom, AuxV, Sb, Obj, Atr, Adv, AtrAdv, AdvAtr, Coord, AtrObj, ObjAtr, AtrAtr, AuxT, AuxR, AuxP, Apos, ExD, AuxC, Atv, AtvV, AuxO, AuxZ, AuxY, AuxG, AuxK, NA
POS	N, A, R, V, D, C, P, J, T, Z, I, NA
SUBPOS	NN, AA, NA, RR, VB, Db, Vp, C=, Dg, PD, Vf, J, J^, P7, P4, PS, Cl, TT, RV, PP, P8, Vs, Cr, AG, Cn, PL, PZ, Vc, AU, PH, Z:, PW, AC, NX, Ca, PQ, P5, PJ, Cv, PK, PE, P1, Vi, P9, A2, CC, P6, Cy, C?, RF, Co, Ve, II, Cd, Ch, J*, AM, Cw, AO, Vt, Vm
is_member	true, false
is_parenthesis	true, false
sempos	n.denot, n.denot.neg, n.pron.def.demon, n.pron.def.pers, n.pron.indef, n.quant.def, adj.denot, adj.pron.def.demon, adj.pron.indef, adj.quant.def, adj.quant.indef, adj.quant.grad, adv.denot.grad.nneg, adv.denot.ngrad.nneg, adv.denot.grad.neg, adv.denot.ngrad.neg, adv.pron.def, adv.pron.indef, v, NA
number	sg, pl, inher, nr, NA
gender	anim, inan, fem, neut, inher, nr, NA
person	1, 2, 3, inher, NA
degcmp	pos, comp, acomp, sup, nr, NA
verbmod	ind, imp, cdn, nr, NA
aspect	proc, cpl, nr, NA
tense	sim, ant, post, nil, NA
numertype	basic, set, kind, ord, frac, NA
indeftype	relat, indef1, indef2, indef3, indef4, indef5, indef6, inter, negat, total1, total2, NA
negation	neg0, neg1, NA
politeness	polite, basic, inher, NA
deontmod	deb, hrt, vol, poss, perm, fac, decl, NA
dispmod	disp1, disp0, nil, NA
resultative	res1, res0, NA
iterativeness	it1, it0, NA

DERIVED FEATURE	POSSIBLE VALUES
is_rightmost	true, false
is_rightside_from_verb	true, false
is_leftside_dependent	true, false
is_embedded_attribute	true, false
has_repeated_lemma	true, false
is_in_canonical_order	true, false
is_weak_pronoun	true, false
is_indexical_expression	true, false
is_pronoun_with_general_meaning	true, false
is_strong_pronoun_with_no_prep	true, false

Using Semantic Relations to Refine Coreference Decisions

Heng Ji **David Westbrook** **Ralph Grishman**
Department of Computer Science
New York University
New York, NY, 10003, USA
hengji@cs.nyu.edu westbroo@cs.nyu.edu grishman@cs.nyu.edu

Abstract

We present a novel mechanism for improving reference resolution by using the output of a relation tagger to rescore coreference hypotheses. Experiments show that this new framework can improve performance on two quite different languages -- English and Chinese.

1 Introduction

Reference resolution has proven to be a major obstacle in building robust systems for information extraction, question answering, text summarization and a number of other natural language processing tasks.

Most reference resolution systems use representations built out of the lexical and syntactic attributes of the noun phrases (or “mentions”) for which reference is to be established. These attributes may involve string matching, agreement, syntactic distance, and positional information, and they tend to rely primarily on the immediate context of the noun phrases (with the possible exception of sentence-spanning distance measures such as Hobbs distance). Though gains have been made with such methods (Tetreault 2001; Mitkov 2000; Soon et al. 2001; Ng and Cardie 2002), there are clearly cases where this sort of local information will not be sufficient to resolve coreference correctly.

Coreference is by definition a semantic relationship: two noun phrases corefer if they both refer to the same real-world entity. We should therefore expect a successful coreference system to exploit world knowledge, inference, and other forms of semantic information in order to resolve hard cases. If, for example, two nouns refer to people who work for two different organizations, we want our system to infer that these noun phrases cannot corefer. Further progress will likely be aided by flexible frameworks for representing and using the information provided by this kind of semantic relation between noun phrases.

This paper tries to make a small step in that direction. It describes a robust reference resolver that incorporates a broad range of semantic information in a general news domain. Using an ontology that describes relations between entities (the Automated Content Extraction program[1] relation ontology) along with a training corpus annotated for relations under this ontology, we first train a classifier for identifying relations. We then apply the output of this relation tagger to the task of reference resolution.

The rest of this paper is structured as follows. Section 2 briefly describes the efforts made by previous researchers to use semantic information in reference resolution. Section 3 describes our own method for incorporating document-level semantic context into coreference decisions. We propose a representation of semantic context that isolates a particularly informative structure of interaction between semantic relations and coreference. Section 4 explains in detail our strategies for using relation information to modify coreference decisions, and the linguistic intuitions behind these strategies. Section 5 then presents the system architectures and algorithms we use to incorporate relational information into reference resolution.

[1] The ACE task description can be found at http://www.itl.nist.gov/iad/894.01/tests/ace/ and the ACE guidelines at http://www.ldc.upenn.edu/Projects/ACE/

Section 6 presents the results of experiments on both English and Chinese test data. Section 7 presents our conclusions and directions for future work.

2 Prior Work

Much of the earlier work in anaphora resolution (from the 1970's and 1980's, in particular) relied heavily on deep semantic analysis and inference procedures (Charniak 1972; Wilensky 1983; Carbonell and Brown 1988; Hobbs et al. 1993). Using these methods, researchers were able to give accounts of some difficult examples, often by encoding quite elaborate world knowledge. Capturing sufficient knowledge to provide adequate coverage of even a limited but realistic domain was very difficult. Applying these reference resolution methods to a broad domain would require a large scale knowledge-engineering effort.

The focus for the last decade has been primarily on broad coverage systems using relatively shallow knowledge, and in particular on corpus-trained statistical models. Some of these systems attempt to apply shallow semantic information. (Ge et al. 1998) incorporate gender, number, and animaticity information into a statistical model for anaphora resolution by gathering coreference statistics on particular nominal-pronoun pairs. (Tetreault and Allen 2004) use a semantic parser to add semantic constraints to the syntactic and agreement constraints in their Left-Right Centering algorithm. (Soon et al. 2001) use WordNet to test the semantic compatibility of individual noun phrase pairs. In general these approaches do not explore the possibility of exploiting the global semantic context provided by the document as a whole.

Recently Bean and Riloff (2004) have sought to acquire automatically some semantic patterns that can be used as contextual information to improve reference resolution, using techniques adapted from information extraction. Their experiments were conducted on collections of texts in two topic areas (terrorism and natural disasters).

3 Relational Model of Semantic Context

Our central goal is to model semantic and coreference structures in such a way that we can take advantage of a semantic context larger than the individual noun phrase when making coreference decisions. Ideally, this model should make it possible to pick out important features in the context and to distinguish useful signals from background noise. It should, for example, be able to represent such basic relational facts as whether the (possibly identical) people referenced by two noun phrases work in the same organization, whether they own the same car, etc. And it should be able to use this information to resolve references even when surface features such as lexical or grammatical attributes are imperfect or fail altogether.

In this paper we present a Relational Coreference Model (abbreviated as RCM) that makes progress toward these goals. To represent semantic relations, we use an ontology (the ACE 2004 relation ontology) that describes 7 main types of relations between entities and 23 subtypes (Table 1).[2] These relations prove to be more reliable guides for coreference than simple lexical context or even tests for the semantic compatibility of heads and modifiers. The process of tagging relations implicitly selects relevant items of context and abstracts raw lists of modifiers into a representation that is deeper, but still relatively lightweight.

Relation Type	Example
Agent-Artifact (ART)	Rubin Military Design, the **makers** of the **Kursk**
Discourse (DISC)	**each** of **whom**
Employment/ Membership (EMP-ORG)	Mr. Smith, a senior **programmer** at **Microsoft**
Place-Affiliation (GPE-AFF)	**Salzburg** Red Cross **officials**
Person-Social (PER-SOC)	**relatives** of the **dead**
Physical (PHYS)	a **town** some 50 miles south of **Salzburg**
Other-Affiliation (Other-AFF)	**Republican senators**

Table 1. Examples of the ACE Relation Types

Given these relations we can define a semantic context for a candidate mention coreference pair (Mention 1b and Mention 2b) using the structure

[2] See http://www.ldc.upenn.edu/Projects/ACE/docs/EnglishRDCV4-3-2.PDF for a more complete description of ACE 2004 relations.

depicted in Figure 1. If both mentions participate in relations, we examine the types and directions of their respective relations as well as whether or not their relation partners (Mention 1a and Mention 2a) corefer. These values (which correspond to the edge labels in Figure 1) can then be factored into a coreference prediction. This RCM structure assimilates relation information into a coherent model of semantic context.

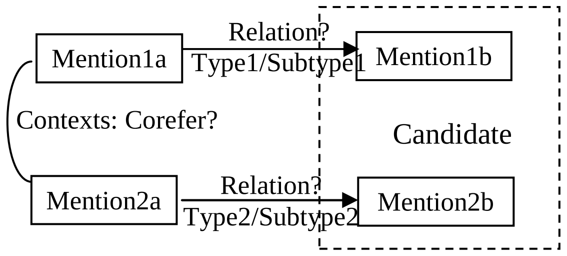


Figure 1. The RCM structure

4 Incorporating Relations into Reference Resolution

Given an instance of the RCM structure, we need to convert it into semantic knowledge that can be applied to a coreference decision. We approach this problem by constructing a set of RCM patterns and evaluating the accuracy of each pattern as positive or negative evidence for coreference. The resulting knowledge sources fall into two categories: rules that improve precision by pruning incorrect coreference links between mentions, and rules that improve recall by recovering missed links.

To formalize these relation patterns, based on Figure 1, we define the following clauses:

A: RelationType1 = RelationType2
B: RelationSubType1 = RelationSubType2
C: Two Relations have the same direction
Same_Relation: $A \wedge B \wedge C$
CorefA: Mention1a and Mention2a corefer
CorefBMoreLikely: Mention1b and Mention2b are more likely to corefer
CorefBLessLikely: Mention1b and Mention2b are less likely to corefer

From these clauses we can construct the following plausible inferences:

Rule (1)
$Same\_Relation \wedge \neg CorefA \Rightarrow CorefBLessLikely$
Rule (2)
$\neg Same\_Relation \wedge CorefA \Rightarrow CorefBLessLikely$
Rule (3)
$Same\_Relation \wedge CorefA \Rightarrow CorefBMoreLikely$

Rule (1) and (2) can be used to prune coreference links that simple string matching might incorrectly assert; and (3) can be used to recover missed mention pairs.

The accuracy of Rules (1) and (3) varies depending on the type and direction of the particular relation shared by the two noun phrases. For example, if Mention1a and Mention 2a both refer to the same nation, and Mentions 1b and 2b participate in citizenship relations (GPE-AFF) with Mentions 1a and 2a respectively, we should not necessarily conclude that 1b and 2b refer to the same person. If 1a and 2a refer to the same person, however, and 1b and 2b are nations in citizenship relations with 1a and 2a, then it would indeed be the rare case in which 1b and 2b refer to two different nations. In other words, the relation of a nation to its citizens is one-to-many.

Our system learns broad restrictions like these by evaluating the accuracy of Rules (1) and (3) when they are instantiated with each possible relation type and direction and used as weak classifiers. For each such instantiation we use cross-validation on our training data to calculate a reliability weight defined as:

$$\frac{|\text{Correct decisions by rule for given instance}|}{|\text{Total applicable cases for given instance}|}$$

We count the number of correct decisions for a rule instance by taking the rule instance as the only source of information for coreference resolution and making only those decisions suggested by the rule's implication (interpreting CorefBMoreLikely as an assertion that mention 1b and mention 2b do in fact corefer, and interpreting CorefBLessLikely as an assertion that they do not corefer).

Every rule instance with a reliability weight of 70% or greater is retained for inclusion in the final system. Rule (2) cannot be instantiated with a single type because it requires that the two relation types be different, and so we do not perform this filtering for Rule (2) (Rule (2) has 97% accuracy across all relation types).

This procedure yields 58 reliable (reliability weight > 70%) type instantiations of Rule (1) and (3), in addition to the reliable Rule 2. We can recover an additional 24 reliable rules by conjoining additional boolean tests to less reliable rules. Tests include equality of mention heads, substring matching, absence of temporal key words such as “current” and “former,” number agreement, and high confidence for original coreference decisions (Mention1b and Mention2b). For each rule below the reliability threshold, we search for combinations of 3 or fewer of these restrictions until we achieve reliability of 70% or we have exhausted the search space.

We give some examples of particular rule instances below.

Example for Rule (1)

> Bush campaign officials ... decided to tone down a post-debate rally, and were even considering canceling it.
> …
> The Bush and Gore campaigns did not talk to each other directly about the possibility of postponement, but went through the debate commission's director, Janet Brown...Eventually, Brown recommended that the debate should go on, and neither side objected, according to campaign officials.

Two mentions that do not corefer share the same nominal head (“officials”). We can prune the coreference link by noting that both occurrences of “officials” participate in an Employee-Organization (EMP-ORG) relation, while the Organization arguments of these two relation instances do not corefer (because the second occurrence refers to officials from both campaigns).

Example for Rule (2)

> Despite the increases, college remains affordable and a good investment, said College Board President Gaston Caperton in a statement with the surveys. …
> A majority of students need grants or loans -- or both -- but their exact numbers are unknown, a College Board spokesman said.

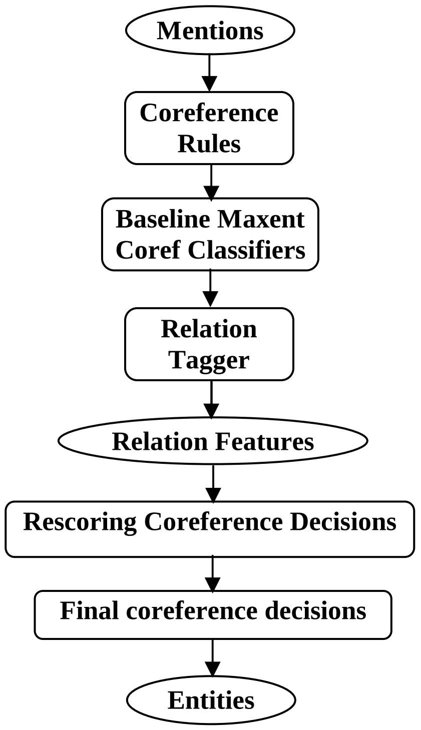
“Gaston Caperton” stands in relation EMP-ORG/Employ-Executive with “College Board”, while "a College Board spokesman" is in relation EMP-ORG/Employ-Staff with the same organization. We conclude that “Gaston Caperton” does not corefer with "spokesman."

Example for Rule (3)

> In his foreign policy debut for Syria, this Sunday Bashar Assad met Sunday with Egyptian President Hosni Mubarak in talks on Mideast peace and the escalating violence in the Palestinian territories.
> …
> The Syrian leader's visit came on a fourth day of clashes that have raged in the West Bank, Gaza Strip and Jerusalem……

If we have detected a coreference link between “Syria” and “Syrian,” as well as EMP-ORG/ Employ-Executive relations between this country and two noun phrases “Bashar Assad” and “leader”, it is likely that the two mentions both refer to the same person. Without this inference, a resolver might have difficulty detecting this coreference link.

5 Algorithms



Mentions
Coreference Rules
Baseline Maxent Coref Classifiers
Relation Tagger
Relation Features
Rescoring Coreference Decisions
Final coreference decisions
Entities

Figure 2. System Pipeline (Test Procedure)

In this section we will describe our algorithm for incorporating semantic relation information from the RCM into the reference resolver. In a nutshell, the system applies a baseline statistical resolver to generate multiple coreference hypotheses, applies a relation tagger to acquire relation information, and uses the relation information to rescore the coreference hypotheses. This general system architecture is shown in Figure 2.

In section 5.1 below we present our baseline coreference system. In Section 5.2 we describe a system that combines the output of this baseline system with relation information to improve performance.

5.1 Baseline System

Baseline reference resolver

As the first stage in the resolution process we apply a baseline reference resolver that uses no relation information at all. This baseline resolver goes through two successive stages.

First, high-precision heuristic rules make some positive and negative reference decisions. Rules include simple string matching (e.g., names that match exactly are resolved), agreement constraints (e.g., a nominal will never be resolved with an entity that doesn't agree in number), and reliable syntactic cues (e.g., mentions in apposition are resolved). When such a rule applies, it assigns a confidence value of 1 or 0 to a candidate mention-antecedent pair.

The remaining pairs are assigned confidence values by a collection of maximum entropy models. Since different mention types have different coreference problems, we separate the system into different models for names, nominals, and pronouns. Each model uses a distinct feature set, and for each instance only one of these three models is used to produce a probability that the instance represents a correct resolution of the mention. When the baseline is used as a standalone system, we apply a threshold to this probability: if some resolution has a confidence above the threshold, the highest confidence resolution will be made. Otherwise the mention is assumed to be the first mention of an entity. When the baseline is used as a component of the system depicted in figure 2, the confidence value is passed on to the rescoring stage described in 5.2 below.

Both the English and the Chinese coreference models incorporate features representing agreement of various kinds between noun phrases (number, gender, humanness), degree of string similarity, synonymy between noun phrase heads, measures of distance between noun phrases (such as the number of intervening sentences), the presence or absence of determiners or quantifiers, and a wide variety of other properties.

Relation tagger

The relation tagger uses a K-nearest-neighbor algorithm. We consider a mention pair as a possible instance of a relation only when: (1) there is at most one other mention between their heads, and (2) the coreference probability produced for the pair by the baseline resolver is lower than a threshold. Each training / test example consists of the pair of mentions and the sequence of intervening words. We defined a distance metric between two examples based on:

- ❑ whether the heads of the mentions match
- ❑ whether the ACE types of the heads of the mentions match (for example, both are people or both are organizations)
- ❑ whether the intervening words match

To tag a test example, we find the k nearest training examples, use the distance to weight each neighbor, and then select the most heavily weighted class in the weighted neighbor set.

Name tagger and noun phrase chunker

Our baseline name tagger consists of a HMM tagger augmented with a set of post-processing rules. The HMM tagger generally follows the Nymble model (Bikel et al. 1997), but with a larger number of states (12 for Chinese, 30 for English) to handle name prefixes and suffixes, and, for Chinese, transliterated foreign names separately. For Chinese it operates on the output of a word segmenter from Tsinghua University. Our nominal mention tagger (noun phrase chunker) is a maximum entropy tagger trained on treebanks from the University of Pennsylvania.

5.2 Rescoring stage

To incorporate information from the relation tagger into the final coreference decision, we split the maxent classification into two stages. The first stage simply applies the baseline maxent models, without any relation information, and produces a probability of coreference. This probability becomes a feature in the second (rescoring) stage of maxent classification, together with features representing the relation knowledge sources. If a high reliability instantiation of one of the RCM rules (as defined in section 4 above) applies to a given mention-antecedent pair, we include the following features for that pair: the type of the RCM rule, the reliability of the rule instantiation, the relation type and subtype, the direction of the relation, and the tokens for the two mentions.

The second stage helps to increase the margin between correct and incorrect links and so effects better disambiguation. See figure 3 below for a more detailed description of the training and testing processes.

Training

1. Calculate reliability weights of relation knowledge sources using cross-validation (for each of k divisions of training data, train relation tagger on k – 1 divisions, tag relations in remaining division and compute reliability of each relation knowledge source using this division).
2. Use high reliability relation knowledge sources to generate relation features for 2nd stage Maxent training data.
3. Apply baseline coreference resolver to 2nd stage training data.
4. Using output of both 2 and 3 as features, train 2nd stage Maxent resolver.

Test

1. Tag relations.
2. Convert relation knowledge sources into features for second stage Maxent models.
3. Use baseline Maxent models to get coreference probabilities for use as features in second stage Maxent models.
4. Using output of 2 and 3 as features for 2nd stage Maxent model, apply 2nd stage resolver to make final coreference decisions.

Figure 3. Training and Testing Processes

6 Evaluation Results

6.1 Corpora

We evaluated our system on two languages: English and Chinese. The following are the training corpora used for the components in these two languages.

English

For English, we trained the baseline maxent coreference model on 311 newswire and newspaper texts from the ACE 2002 and ACE 2003 training corpora. We trained the relation tagger on 328 ACE 2004 texts. We used 126 newswire texts from the ACE 2004 data to train the English second-stage model, and 65 newswire texts from the ACE 2004 evaluation set as a test set for the English system.

Chinese

For Chinese, the baseline reference resolver was trained on 767 texts from ACE 2003 and ACE 2004 training data. Both the baseline relation tagger and the rescoring model were trained on 646 texts from ACE 2004 training data. We used 100 ACE texts for a final blind test.

6.2 Experiments

We used the MUC coreference scoring metric (Vilain et al 1995) to evaluate[3] our systems.

To establish an upper limit for the possible improvement offered by our models, we first did experiments using perfect (hand-tagged) mentions and perfect relations as inputs. The algorithms for

[3] In our scoring, we use the ACE keys and only score mentions which appear in both the key and system response. This therefore includes only mentions identified as being in the ACE semantic categories by both the key and the system response. Thus these scores cannot be directly compared against coreference scores involving all noun phrases. (Ng 2005) applies another variation on the MUC metric to several systems tested on the ACE data by scoring all response mentions against all key mentions. For coreference systems that don't restrict themselves to mentions in the ACE categories (or that don't succeed in so restricting themselves), this scoring method could lead to some odd effects. For example, systems that recover more correct links could be penalized for this greater recall because all links involving non-ACE mentions will be incorrect according to the ACE key. For the sake of comparison, however, we present here English system results measured according to this metric: On newswire data, our baseline had an F of 62.8 and the rescoring method had an F of 64.2. Ng's best F score (on newspaper data) is 69.3. The best F score of the (Ng and Cardie 2002) system (also on newspaper data) is 62.1. On newswire data the (Ng 2005) system had an F score of 54.7 and the (Ng and Cardie 2002) system had an F score of 50.1. Note that Ng trained and tested these systems on different ACE data sets than those we used for our experiments.

these experiments are identical to those described above except for the omission of the relation tagger training. Tables 2 and 3 show the performance of the system for English and Chinese.

Performance	Recall	Precision	F-measure
Baseline	74.5	86.6	80.1
Rescoring	78.3	87.0	82.4

Table 2. Performance of English system with perfect mentions and perfect relations

Performance	Recall	Precision	F-measure
Baseline	87.5	83.2	85.3
Rescoring	88.8	84.7	86.7

Table 3. Performance of Chinese system with perfect mentions and perfect relations

We can see that the relation information provided some improvements for both languages. Relation information increased both recall and precision in both cases.

We then performed experiments to evaluate the impact of coreference rescoring when used with mentions and relations produced by the system. Table 4 and Table 5 list the results.[4]

Performance	Recall	Precision	F-measure
Baseline	77.2	87.3	81.9
Rescoring	80.3	87.5	83.7

Table 4. Performance of English system with system mentions and system relations

Performance	Recall	Precision	F-measure
Baseline	75.0	76.3	75.6
Rescoring	76.1	76.5	76.3

Table 5. Chinese system performance with system mentions and system relations

[4] Note that, while English shows slightly less relative gain from rescoring when using system relations and mentions, all of these scores are higher than the perfect mention/perfect relation scores. This increase may be a byproduct of the fact that the system mention tagger output contains almost 8% fewer scoreable mentions than the perfect mention set (see footnote 3). With a difference of this magnitude, the particular mention set selected can be expected to have a sizable impact on the final scores.

The improvement provided by rescoring in trials using mentions and relations detected by the system is considerably less than the improvement in trials using perfect mentions and relations, particularly for Chinese. The performance of our relation tagger is the most likely cause for this difference. We would expect further gain after improving the relation tagger.

A sign test applied to a 5-way split of each of the test corpora indicated that for both languages, for both perfect and system mentions/relations, the system that exploited relation information significantly outperformed the baseline (at the 95% confidence level, judged by F measure).

6.3 Error Analysis

Errors made by the RCM rules reveal both the drawbacks of using a lightweight semantic representation and the inherent difficulty of semantic analysis. Consider the following instance:

> Card's interest in politics began when he became **president** of **the class of 1965 at Holbrook High School**…In 1993, he became **president** and chief executive of **the American Automobile Manufacturers Association**, where he oversaw the lobbying against tighter fuel-economy and air pollution regulations for automobiles…

The two occurrences of “president” should corefer even though they have EMP-ORG/Employ-Executive relations with two different organizations. The relation rule (Rule 1) fails here because it doesn't take into account the fact that relations change over time (in this case, the same person filling different positions at different times). In these and other cases, a little knowledge is a dangerous thing: a more complete schema might be able to deal more thoroughly with temporal and other essential semantic dimensions.

Nevertheless, performance improvements indicate that the rewards of the RCM’s simple semantic representation outweigh the risks.

7 Conclusion and Future Work

We have outlined an approach to improving reference resolution through the use of semantic relations, and have described a system which can exploit these semantic relations effectively. Our experiments on English and Chinese data showed that these small inroads into semantic territory do indeed offer performance improvements. Furthermore, the method is low-cost and not domain-specific.

These experiments also suggest that some gains can be made through the exploration of new architectures for information extraction applications. The "resolve coreference, tag relations, resolve coreference" procedure described above could be seen as one and a half iterations of a "resolve coreference then tag relations" loop. Seen in this way, the system poses the question of whether further gains could be made by pushing the iterative approach further. Perhaps by substituting an iterative procedure for the pipeline architecture's linear sequence of stages we can begin to address the knotty, mutually determining nature of the interaction between semantic relations and coreference relations. This approach could be applied more broadly, to different NLP tasks, and also more deeply, going beyond the simple one-and-a-half-iteration procedure we present here. Ultimately, we would want this framework to boost the performance of each component automatically and significantly.

We also intend to extend our method both to cross-document relation detection and to event detection.

Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency under Grant N66001-04-1-8920 from SPAWAR San Diego, and by the National Science Foundation under Grant 03-25657. This paper does not necessarily reflect the position or the policy of the U.S. Government.

References

David Bean, Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. *Proc. HLT-NAACL 2004,* pp. 297-304.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: A high-performance learning name-finder. *Proc. Fifth Conf. on Applied Natural Language Processing,* Washington, D.C., pp. 194-201.

Carbonell, Jaime and Ralf Brown. 1988. Anaphora resolution: A multi-strategy approach. *Proc. COLING 1988,* pp.96-101

Eugene Charniak. 1972. Toward a model of children's story comprehension. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Niyu Ge, John Hale and Eugene Charniak. 1998. A statistical approach to anaphora resolution. *Proc. the Sixth Workshop on Very Large Corpora.*

Jerry Hobbs, Mark Stickel, Douglas Appelt and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence,* 63, pp. 69-142.

Ruslan Mitkov. 2000. Towards a more consistent and comprehensive evaluation of anaphora resolution algorithms and systems. *Proc. 2nd Discourse Anaphora and Anaphora Resolution Colloquium,* pp. 96-107

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. *Proc. ACL 2002*, pp.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,* Volume 27, Number 4, pp. 521-544

Joel R. Tetreault. 2001. A corpus-based evaluation of centering and pronoun resolution. *Computational Lin*guistics, Volume 27, Number 4, pp. 507-520

Joel R. Tetreault and James Allen. 2004. Semantics, Dialogue, and Pronoun Resolution. *Proc. CATALOG '04* Barcelona, Spain.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. *Proc. the 6th Message Understanding Conference (MUC-6).* San Mateo, Cal. Morgan Kaufmann.

Robert Wilensky. 1983. *Planning and Understanding.* Addison-Wesley.

On Coreference Resolution Performance Metrics

Xiaoqiang Luo
1101 Kitchawan Road, Room 23-121
IBM T.J. Wastson Research Center
Yorktown Heights, NY 10598, U.S.A.
`xiaoluo@us.ibm.com`

Abstract

The paper proposes a Constrained Entity-Alignment F-Measure (CEAF) for evaluating coreference resolution. The metric is computed by aligning reference and system entities (or coreference chains) with the constraint that a system (reference) entity is aligned with at most one reference (system) entity. We show that the best alignment is a maximum bipartite matching problem which can be solved by the Kuhn-Munkres algorithm. Comparative experiments are conducted to show that the widely-known MUC F-measure has serious flaws in evaluating a coreference system. The proposed metric is also compared with the ACE-Value, the official evaluation metric in the Automatic Content Extraction (ACE) task, and we conclude that the proposed metric possesses some properties such as symmetry and better interpretability missing in the ACE-Value.

1 Introduction

A working definition of coreference resolution is partitioning the noun phrases we are interested in into equivalence classes, each of which refers to a physical entity. We adopt the terminologies used in the Automatic Content Extraction (ACE) task (NIST, 2003a) and call each individual phrase a *mention* and equivalence class an *entity*. For example, in the following text segment,

> (1): "The American Medical Association voted yesterday to install the heir apparent as its president-elect, rejecting a strong, upstart challenge by a district doctor who argued that the nation's largest physicians' group needs stronger ethics and new leadership."

mentions are underlined, "American Medical Association", "its" and "group" refer to the same organization (object) and they form an entity. Similarly, "the heir apparent" and "president-elect" refer to the same person and they form another entity. It is worth pointing out that the entity definition here is different from what used in the Message Understanding Conference (MUC) task (MUC, 1995; MUC, 1998) – ACE entity is called coreference chain or equivalence class in MUC, and ACE mention is called entity in MUC.

An important problem in coreference resolution is how to evaluate a system's performance. A good performance metric should have the following two properties:

- Discriminativity: This refers to the ability to differentiate a good system from a bad one. While this criterion sounds trivial, not all performance metrics used in the past possess this property.

- Interpretability: A good metric should be easy to interpret. That is, there should be an intuitive sense of how good a system is when a metric suggests that a certain percentage of coreference results are correct. For example, when a metric reports 95% or above correct for a system, we would expect that the vast majority of mentions are in right entities or coreference chains.

A widely-used metric is the link-based F-measure (Vilain et al., 1995) adopted in the MUC task. It is computed by first counting the number of common links between the reference (or "truth") and the system output (or "response"); the link precision is the number of common links divided by the number of links in the system output, and the link recall is the number of common links divided by the number of links in the reference. There are known problems associated with the link-based F-measure. First, it ignores single-mention entities since no link can be found in these entities; Second, and more importantly, it fails to distinguish system outputs with different qualities: the link-based F-measure intrinsically favors systems producing fewer entities, and may result in higher F-measures for worse systems. We will revisit these issues in Section 3.

To counter these shortcomings, Bagga and Baldwin (1998) proposed a B-cubed metric, which first computes a precision and recall for each individual mention, and then takes the weighted sum of these individual precisions and recalls as the final metric. While the B-cubed metric fixes some of the shortcomings of the MUC F-measure, it has its own problems: for example, the mention precision/recall is computed by comparing entities containing the mention and therefore an entity can be used more than once. The implication of this drawback will be revisited in Section 3.

In the ACE task, a value-based metric called ACE-value (NIST, 2003b) is used. The ACE-value is computed by counting the number of false-alarm, the number of miss, and the number of mistaken entities. Each error is associated with a cost factor that depends on things such as entity type (e.g., "LOCATION", "PERSON"), and mention level (e.g., "NAME," "NOMINAL," and "PRONOUN"). The total cost is the sum of the three costs, which is then normalized against the cost of a nominal system that does not output any entity. The ACE-value is finally computed by subtracting the normalized cost from 1. A perfect coreference system will get a 100% ACE-value while a system outputs no entities will get a 0 ACE-value. A system outputting many erroneous entities could even get negative ACE-value. The ACE-value is computed by aligning entities and thus avoids the problems of the MUC F-measure. The ACE-value is, however, hard to interpret: a system with 90% ACE-value does not mean that 90% of system entities or mentions are correct, but that the cost of the system, relative to the one outputting no entity, is 10%.

In this paper, we aim to develop an evaluation metric that is able to measure the quality of a coreference system – that is, an intuitively better system would get a higher score than a worse system, and is easy to interpret. To this end, we observe that coreference systems are to recognize *entities* and propose a metric called Constrained Entity-Aligned F-Measure (CEAF). At the core of the metric is the optimal one-to-one map between subsets of reference and system entities: system entities and reference entities are aligned by maximizing the total entity similarity under the constraint that a reference entity is aligned with at most one system entity, and vice versa. Once the total similarity is defined, it is straightforward to compute recall, precision and F-measure. The constraint imposed in the entity alignment makes it impossible to "cheat" the metric: a system outputting too many entities will be penalized in precision while a system outputting two few entities will be penalized in recall. It also has the property that a perfect system gets an F-measure 1 while a system outputting no entity or no common mentions gets an F-measure 0. The proposed CEAF has a clear meaning: for mention-based CEAF, it reflects the percentage of mentions that are in the correct entities; For entity-based CEAF, it reflects the percentage of correctly recognized entities.

The rest of the paper is organized as follows. In Section 2, the Constrained Entity-Alignment F-Measure is presented in detail: the constraint entity alignment can be represented by a bipartite graph and the optimal alignment can be found by the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957). We also present two entity-pair similarity measures that can be used in CEAF: one is the absolute number of common mentions between two entities, and the other is a "local" mention F-measure between two entities. The two measures lead to the mention-based and entity-based CEAF, respectively. In Section 3, we compare the proposed metric with the MUC link-based metric and ACE-value on both artificial and real data, and point out the problems of the MUC F-measure.

2 Constrained Entity-Alignment F-Measure

Some notations are needed before we present the proposed metric and the algorithm to compute the metric.

Let reference entities in a document d be

$$\mathcal{R}(d) = \{R_i : i = 1, 2, \cdots, |\mathcal{R}(d)|\},$$

and system entities be

$$\mathcal{S}(d) = \{S_i : i = 1, 2, \cdots, |\mathcal{S}(d)|\}.$$

To simplify typesetting, we will omit the dependency on d when it is clear from context, and write $\mathcal{R}(d)$ as $\mathcal{R}$ and $\mathcal{S}(d)$ as $\mathcal{S}$.

Let

$$\begin{aligned} m &= \min\{|\mathcal{R}|, |\mathcal{S}|\} \\ M &= \max\{|\mathcal{R}|, |\mathcal{S}|\}, \end{aligned}$$

and let $\mathcal{R}_m \subset \mathcal{R}$ and $\mathcal{S}_m \subset \mathcal{S}$ be any subsets with m entities. That is, $|\mathcal{R}_m| = m$ and $|\mathcal{S}_m| = m$. Let $G(\mathcal{R}_m, \mathcal{S}_m)$ be the set of one-to-one entity maps from $\mathcal{R}_m$ to $\mathcal{S}_m$, and G_m be the set of all possible one-to-one maps between the size-m subsets of $\mathcal{R}$ and $\mathcal{S}$. Or

$$\begin{aligned} G(\mathcal{R}_m, \mathcal{S}_m) &= \{g : \mathcal{R}_m \mapsto \mathcal{S}_m\}, \\ G_m &= \cup_{(\mathcal{R}_m, \mathcal{S}_m)} G(\mathcal{R}_m, \mathcal{S}_m). \end{aligned}$$

The requirement of one-to-one map means that for any $g \in G(\mathcal{R}_m, \mathcal{S}_m)$, and any $R \in \mathcal{R}_m$ and $R' \in \mathcal{R}_m$, we have that $R \neq R'$ implies that $g(R) \neq g(R')$, and $g(R) \neq g(R')$ implies that $R \neq R'$. Clearly, there are $m!$ one-to-one maps from $\mathcal{R}_m$ to $\mathcal{S}_m$ (or $|G(\mathcal{R}_m, \mathcal{S}_m)| = m!$), and $|G_m| = \binom{M}{m} m!$.

Let $\phi(R, S)$ be a "similarity" measure between two entities R and S. $\phi(R, S)$ takes non-negative value: zero value means that R and S have nothing in common. For example, $\phi(R,S)$ could be the number of common mentions shared by R and S, and $\phi(R,R)$ the number of mentions in entity R.

For any $g \in G_m$, the total similarity $\Phi(g)$ for a map g is the sum of similarities between the aligned entity pairs: $\Phi(g) = \sum_{R \in \mathcal{R}_m} \phi(R, g(R))$. Given a document d, and its reference entities $\mathcal{R}$ and system entities $\mathcal{S}$, we can find the best alignment maximizing the total similarity:

$$\begin{aligned} g^* &= \arg\max_{g \in G_m} \Phi(g) \\ &= \arg\max_{g \in G_m} \sum_{R \in \mathcal{R}_m} \phi(R, g(R)). \end{aligned} \tag{1}$$

Let $\mathcal{R}_m^*$ and $\mathcal{S}_m^* = g^*(\mathcal{R}_m^*)$ denote the reference and system entity subsets where g^* is attained, respectively. Then the maximum total similarity is

$$\Phi(g^*) = \sum_{R \in \mathcal{R}_m^*} \phi(R, g^*(R)). \tag{2}$$

If we insist that $\phi(R,S) = 0$ whenever R or S is empty, then the non-negativity requirement of $\phi(R,S)$ makes it unnecessary to consider the possibility of mapping one entity to an empty entity since the one-to-one map maximizing $\Phi(g)$ must be in G_m.

Since we can compute the entity self-similarity $\phi(R,R)$ and $\phi(S,S)$ for any $R \in \mathcal{R}$ and $S \in \mathcal{S}$ (i.e., using the identity map), we are now ready to define the precision, recall and F-measure as follows:

$$p = \frac{\Phi(g^*)}{\sum_i \phi(S_i, S_i)} \tag{3}$$

$$r = \frac{\Phi(g^*)}{\sum_i \phi(R_i, R_i)} \tag{4}$$

$$F = \frac{2pr}{p+r}. \tag{5}$$

The optimal alignment g^* involves only $m = \min\{|\mathcal{R}|, |\mathcal{S}|\}$ reference and system entities, and entities not aligned do not get credit. Thus the F-measure (5) penalizes a coreference system that proposes too many (i.e., lower precision) or too few entities (i.e., lower recall), which is a desired property.

In the above discussion, it is assumed that the similarity measure $\phi(R,S)$ is computed for all entity pair (R,S). In practice, computation of $\phi(R,S)$ can be avoided if it is clear that R and S have nothing in common (e.g., if no mention in R and S overlaps, then $\phi(R,S) = 0$). These entity pairs are not linked and they will not be considered when searching for the optimal alignment. Consequently the optimal alignment could involve less than m reference and system entities. This can speed up considerably the F-measure computation when the majority of entity pairs have zero similarity. Nevertheless, summing over m entity pairs in the general formulae (2) does not change the optimal total similarity between $\mathcal{R}$ and $\mathcal{S}$ and hence the F-measure.

In formulae (3)-(5), there is only one document in the test corpus. Extension to corpus with multiple test documents is trivial: just accumulate statistics on the per-document basis for both denominators and numerators in (3) and (4), and find the ratio of the two.

So far, we have tacitly kept abstract the similarity measure $\phi(R,S)$ for entity pair R and S. We will defer the discussion of this metric to Section 2.2. Instead, we first present the algorithm computing the F-measure (3)-(5).

2.1 Computing Optimal Alignment and F-measure

A naive implementation of (1) would enumerate all the possible one-to-one maps (or alignments) between size-m (recall that $m = \min\{|\mathcal{R}|, |\mathcal{S}|\}$) subsets of $\mathcal{R}$ and size-m subsets of $\mathcal{S}$, and find the best alignment maximizing the similarity. Since this requires computing the similarities between mM entity pairs and there are $|G_m| = \binom{M}{m} m!$ possible one-to-one maps, the complexity of this implementation is $O(Mm + \binom{M}{m} m!)$. This is not satisfactory even for a document with a moderate number of entities: it will have about 3.6 million operations for $M = m = 10$, a document with only 10 reference and 10 system entities.

Fortunately, the entity alignment problem under the constraint that an entity can be aligned at most once is the classical maximum bipartite matching problem and there exists an algorithm (Kuhn, 1955; Munkres, 1957) (henceforth Kuhn-Munkres Algorithm) that can find the optimal solution in polynomial time. Casting the entity alignment problem as the maximum bipartite matching is trivial: each entity in $\mathcal{R}$ and $\mathcal{S}$ is a vertex and the node pair (R,S), where $R \in \mathcal{R}$, $S \in \mathcal{S}$, is connected by an edge with the weight $\phi(R,S)$. Thus the problem (1) is exactly the maximum bipartite matching.

With the Kuhn-Munkres algorithm, the procedure to compute the F-measure (5) can be described as Algorithm 1.

Algorithm 1 Computing the F-measure (5).

```
Input: reference entities:$\mathcal{R}$;    system entities: $\mathcal{S}$
Output: optimal alignment $g^*$; F-measure (5).
1:Initialize: $g^* = \emptyset$; $\Phi(g^*) = 0$.
2:For $i = 1$ to $|\mathcal{R}|$
3:   For $j = 1$ to $|\mathcal{S}|$
4:       Compute $\phi(R_i, S_j)$.
5:[$g^*$,$\Phi(g^*)$]=KM$\big(\{\phi(R,S) : R \in \mathcal{R}, S \in \mathcal{S}\}\big)$.
6:$\Phi(\mathcal{R}) = \sum_{R \in \mathcal{R}} \phi(R,R)$; $\Phi(\mathcal{S}) = \sum_{S \in \mathcal{S}} \phi(S,S)$.
7:$r = \frac{\Phi(g^*)}{\Phi(\mathcal{R})}$; $p = \frac{\Phi(g^*)}{\Phi(\mathcal{S})}$; $F = \frac{2pr}{p+r}$.
8:return $g^*$ and $F$.
```

The input to the algorithm are reference entities $\mathcal{R}$ and system entities $\mathcal{S}$. The algorithm returns the best one-to-one map g^* and F-measure in equation (5). Loop from line 2 to 4 computes the similarity between all the possible reference and system entity pairs. The complexity of this loop is $O(Mm)$. Line 5 calls the Kuhn-Munkres algorithm, which takes as input the entity-pair scores $\{\phi(R,S)\}$ and outputs the best map g^* and the corresponding total similarity $\Phi(g^*)$. The worst case (i.e., when all entries in $\{\phi(R,S)\}$ are non-zeros) complexity of the Kuhn-Algorithm is $O(Mm^2 \log m)$. Line 6 computes "self-similarity" $\Phi(\mathcal{R})$ and $\Phi(\mathcal{S})$ needed in the F-measure computation at Line 7.

The core of the F-measure computation is the Kuhn-Munkres algorithm at line 5. The algorithm is initially discovered by Kuhn (1955) and Munkres (1957) to solve the matching (a.k.a assignment) problem for square matrices. Since then, it has been extended to rectangular matrices (Bourgeois and Lassalle, 1971) and parallelized (Balas et al., 1991). A recent review can be found in (Gupta and Ying, 1999), which also details the techniques of fast implementation. A short description of the algorithm is included in Appendix for the sake of completeness.

2.2 Entity Similarity Metric

In this section we consider the entity similarity metric $\phi(R,S)$ defined on an entity pair (R,S). It is desirable that $\phi(R,S)$ is large when R and S are "close" and small when R and S are very different. Some straight-forward choices could be

$$\phi_1(R,S) = \{ \begin{array}{ll} 1, & \text{if } R = S \\ 0, & \text{otherwise.} \end{array} \quad (6)$$

$$\phi_2(R,S) = \{ \begin{array}{ll} 1, & \text{if } R \cap S \neq \emptyset \\ 0, & \text{otherwise.} \end{array} \quad (7)$$

(6) insists that two entity are the same if all the mentions are the same, while (7) goes to the other extreme: two entities are the same if they share at least one common mention.

(6) does not offer a good granularity of similarity: For example, if $R = \{a,b,c\}$, and one system response is $S_1 = \{a,b\}$, and the other system response $S_2 = \{a\}$, then clearly S_1 is more similar to R than S_2, yet $\phi(R,S_1) = \phi(R,S_2) = 0$. For the same reason, (7) lacks of the desired discriminativity as well.

From the above argument, it is clear that we want to have a metric that can measure the degree to which two entities are similar, not a binary decision. One natural choice is measuring how many common mentions two entities share, and this can be measured by the absolute number or relative number:

$$\phi_3(R,S) = |R \cap S| \quad (8)$$

$$\phi_4(R,S) = \frac{2|R \cap S|}{|R| + |S|}. \quad (9)$$

Metric (8) simply counts the number of common mentions shared by R and S, while (9) is the mention F-measure between R and S, a relative number measuring how similar R and S are. For the abovementioned example,

$$\begin{aligned} \phi_3(R,S_1) &= \phi_3(\{a,b,c\},\{a,b\}) = 2 \\ \phi_3(R,S_2) &= \phi_3(\{a,b,c\},\{a\}) = 1 \\ \phi_4(R,S_1) &= \phi_4(\{a,b,c\},\{a,b\}) = 0.8 \\ \phi_4(R,S_2) &= \phi_4(\{a,b,c\},\{a\}) = 0.5, \end{aligned}$$

thus both metrics give the desired ranking $\phi_3(R,S_1) > \phi_3(R,S_2)$, $\phi_4(R,S_1) > \phi_4(R,S_2)$.

If $\phi_3(\cdot,\cdot)$ is adopted in Algorithm 1, $\Phi(g^*)$ is the number of total common mentions corresponding to the best one-to-one map g^* while the denominators of (3) and (4) are the number of proposed mentions and the number of system mentions, respectively. The F-measure in (5) can be interpreted as the ratio of mentions that are in the "right" entities. Similarly, if $\phi_4(\cdot,\cdot)$ is adopted in Algorithm 1, the denominators of (3) and (4) are the number of proposed entities and the number of system entities, respectively, and the F-measure in (5) can be understood as the ratio of correct entities. Therefore, (5) is called mention-based CEAF and entity-based CEAF when (8) and (9) are used, respectively.

$\phi_3(\cdot,\cdot)$ and $\phi_4(\cdot,\cdot)$ are two reasonable entity similarity measures, but by no means the only choices. At mention level, partial credit could be assigned to two mentions with different but overlapping spans; or when mention type is available, weights defined on the type confusion matrix can be incorporated. At entity level, entity attributes, if avaiable, can be weighted in the similarity measure as well. For example, ACE data defines three entity classes: NAME, NOMINAL and PRONOUN. Different weights can be assigned to the three classes.

No matter what entity similarity measure is used, it is crucial to have the constraint that the document-level similarity between reference entities and system entities is calculated over the best one-to-one map. We will see examples in Section 3 that misleading results could be produced without the alignment constraint.

Another observation is that the same evaluation paradigm can be used in any scenario that needs to measure the "closeness" between a set of system and reference objects, provided that a similarity between two objects is defined. For example, the 2004 ACE tasks include detecting and recognizing relations in text documents. A relation instance can be treated as an object and the same evaluation paradigm can be applied.

3 Comparison with Other Metrics

In this section, we compare the proposed F-measure with the MUC link-based F-measure (and its variation B-cube F-measure) and the more recent ACE-value. The

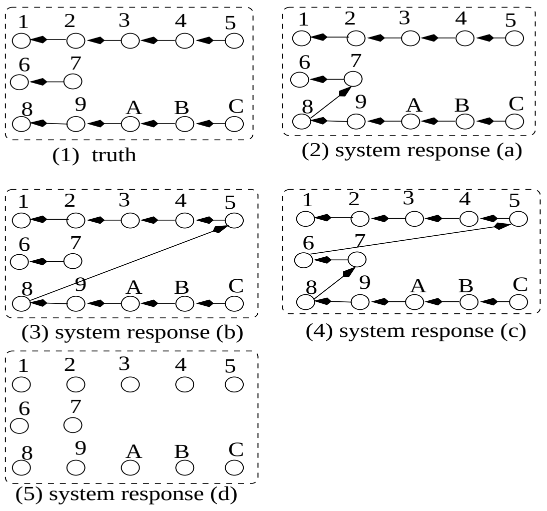


Figure 1: Example entities: (1)truth; (2)system response (a); (3)system response (b); (4)system response (c); (5)system response (d)

proposed metric has fixed problems associated with the MUC and B-cube F-measure, and has better interpretability than the ACE-value.

3.1 Comparison with the MUC F-measure and B-cube Metric on Artificial Data

We use the example in Figure 1 to compare the MUC link-based F-measure, B-cube, and the proposed mention- and entity-based CEAF. In Figure 1, mentions are represented in circles and mentions in an entity are connected by arrows. Intuitively, if each mention is treated equally, the system response (a) is better than the system response (b) since the latter mixes two big entities, $\{1, 2, 3, 4, 5\}$ and $\{8, 9, A, B, C\}$, while the former mixes a small entity $\{6, 7\}$ with one big entity $\{8, 9, A, B, C\}$. System response (b) is clearly better than system response (c) since the latter puts all the mentions into a single entity while (b) has correctly separated the entity $\{6, 7\}$ from the rest. The system response (d) is the worst: the system does not link any mentions and outputs 12 single-mention entities.

Table 1 summarizes various F-measures for system response (a) to (d): the first column contains the indices of the system responses found in Figure 1; the second and third columns are the MUC F-measure and B-cubic F-measure respectively; the last two columns are the proposed CEAF F-measures, using the entity similarity metric $\phi_3(\cdot,\cdot)$ and $\phi_4(\cdot,\cdot)$, respectively.

As shown in Table 1, the MUC link-based F-measure fails to distinguish the system response (a) and the system response (b) as the two are assigned the same F-measure. The system response (c) represents a trivial output: all mentions are put in the same entity. Yet the MUC metric will lead to a 100% recall (9 out of 9 reference links are

System response	MUC	B-cube	CEAF	
			$\phi_3(\cdot,\cdot)$	$\phi_4(\cdot,\cdot)$
(a)	0.947	0.865	0.833	0.733
(b)	0.947	0.737	0.583	0.667
(c)	0.900	0.545	0.417	0.294
(d)	–	0.400	0.250	0.178

Table 1: Comparison of coreference evaluation metrics

correct) and a 81.2% precision (9 out of 11 system links are correct), which gives rise to a 90% F-measure. It is striking that a "bad" system response gets such a high F-measure. Another problem with the MUC link-based metric is that it is not able to handle single-mention entities, as there is no link for a single mention entity. That is why the entry for system response (d) in Table 1 is empty.

B-cube F-measure ranks the four system responses in Table 1 as desired. This is because B-cube metric (Bagga and Baldwin, 1998) is computed based on mentions (as opposed to links in the MUC F-measure). But B-cube uses the same entity "intersecting" procedure found in computing the MUC F-measure (Vilain et al., 1995), and it sometimes can give counter-intuitive results. To see this, let us take a look at recall and precision for system response (c) and (d) for B-cube metric. Notice that all the reference entities are found after intersecting with the system responsce (c): $\{\{1, 2, 3, 4, 5\}, \{6, 7\}, \{8, 9, A, B, C\}\}$. Therefore, B-cube recall is 100% (the corresponding precision is $\frac{1}{12} * (10 * \frac{5}{12} + 2 * \frac{2}{12}) = 0.375$). This is counter-intuitive because the set of reference entities is not a subset of the proposed entities, thus the system response should not have gotten a 100% recall. The same problem exists for the system response (d): it gets a 100% B-cube precision (the corresponding B-cube recall is $\frac{1}{12}(5 * \frac{1}{5} + 2 * \frac{1}{2} + 5 * \frac{1}{5}) = 0.25$), but clearly not all the entities in the system response (d) are correct! These numebrs are summarized in Table 2, where columns with R and P represent recall and precision, respectively.

System response	B-cube		CEAF			
	R	P	ϕ_3-R	ϕ_3-P	ϕ_4-R	ϕ_4-P
(c)	1.0	0.375	0.417	0.417	0.196	0.588
(d)	0.25	1.0	0.250	0.250	0.444	0.111

Table 2: Example of counter-intuitive B-cube recall or precision: system repsonse (c) gets 100% recall (column R) while system repsonse (d) gets 100% precision (column P). The problem is fixed in both CEAF metrics.

The counter-intuitive results associated with the MUC and B-cube F-measures are rooted in the procedure of "intersecting" the reference and system entities, which allows an entity to be used more than once! We will come back to this after discussing the CEAF numbers.

From Table 1, we see that both mention-based (column under $\phi_3(\cdot,\cdot)$) CEAF and entity-based ($\phi_4(\cdot,\cdot)$) CEAF are able to rank the four systems properly: system (a) to (d) are increasingly worse. To see how the CEAF numbers are computed, let us take the system response (a) as an example: first, the best one-one entity map is determined. In this case, the best map is: the reference entity $\{1,2,3,4,5\}$ is aligned to the system entity $\{1,2,3,4,5\}$, the reference entity $\{8,9,A,B,C\}$ is aligned to the system $\{6,7,8,9,A,B,C\}$ and the reference entity $\{6,7\}$ is unaligned. The number of common mentions is therefore 10 which results in a mention-based ($\phi_3(\cdot,\cdot)$) recall $\frac{5}{6}$ and precision $\frac{5}{6}$. Since $\phi_4(\{1,2,3,4,5\},\{1,2,3,4,5\}) = 1$, and $\phi_4(\{8,9,A,B,C\},\{6,7,8,9,A,B,C\}) = \frac{10}{12}$, $\Phi(g^*) = 1 + \frac{10}{12}$ (c.f. equation (4) and (3)), and the entity-based F-measure (c.f. equation (9)) is therefore

$$\frac{2 * (1+\frac{10}{12})}{3+2} = \frac{11}{15} = 0.733.$$

CEAF for other system responses are computed similarly.

CEAF recall and precision breakdown for system (c) and (d) are listed in column 4 through 7 of Table 1. As can be seen, neither mention-based nor entity-based CEAF has the abovementioned problem associated with the B-cube metric, and the recall and precision numbers are more or less compatible with our intuition: for instance, for system (c), based on ϕ_3-CEAF number, we can say that about 41.7% mentions are in the right entity, and based on the ϕ_4-CEAF recall and precision, we can state that about 19.6% of "true" entities are recovered (recall) and about 58.8% of the proposed entities are correct.

A comparison of the procedures of computing the MUC F-measure/B-cube and CEAF reveals that the crucial difference is that the MUC and B-cube F-measure allow an entity to be used multiple times while CEAF insists that entity map be one-to-one. So an entity will never get double credit. Take the system repsonse (c) as an example, intersecting three reference entity in turn with the reference entities produces the same set of reference entities, which leads to 100% recall. In the intersection step, the system entity is effectively used three times. In contrast, the system entity is aligned to only one reference entity when computing CEAF.

3.2 Comparisons On Real Data

3.2.1 MUC F-measure and CEAF

We have seen the different behaviors of the MUC F-measure, B-cube F-measure and CEAF on the artificial data. We now compare the MUC F-measure, CEAF, and ACE-value metrics on real data (compasion between the MUC and B-cube F-measure can be found in (Bagga and Baldwin, 1998)). Comparsion between the MUC F-measure and CEAF is done on the MUC6 coreference test set, while comparison between the CEAF and ACE-value is done on the 2004 ACE data. The setup reflects the fact that the official MUC scorer and ACE scorer run on their own data format and are not easily portable to the other data set. All the experiments in this section are done on true mentions.

Penalty	#sys-ent	MUC-F	ϕ_3-CEAF
-0.6	561	.851	0.750
-0.8	538	.854	0.756
-0.9	529	.853	0.753
-1	515	.853	0.753
-1.1	506	.856	0.764
-1.2	483	.857	**0.768**
-1.4	448	.863	0.761
-1.5	425	.862	0.749
-1.6	411	.864	0.740
-1.7	403	.865	0.741
-10	113	**.902**	0.445

Table 3: MUC F-measure and mention-based CEAF on the official MUC6 test set. The first column contains the penalty value in decreasing order. The second column contains the number of system-proposed entities. The column under MUC-F is the MUC F-measure while ϕ_3-CEAF is the mention-based CEAF.

The coreference system is similar to the one used in (Luo et al., 2004). Results in Table 3 are produced by a system trained on the MUC6 training data and tested on the 30 official MUC6 test documents. The test set contains 460 reference entities. The coreference system uses a penalty parameter to balance miss and false alarm errors: the smaller the parameter, the fewer entities will be generated. We vary the parameter from -0.6 to -10, listed in the first column of Table 3, and compare the system performance measured by the MUC F-measure and the proposed mention-based CEAF.

As can be seen, the mention-based CEAF has a clear maximum when the number of proposed entities is close to the truth: at the penlaty value -1.2, the system produces 483 entities, very close to 460, and the ϕ_3-CEAF achieves the maximum 0.768. In contrast, the MUC F-measure increases almost monotonically as the system proposes fewer and fewer entities. In fact, the best system according to the MUC F-measure is the one proposing only 113 entities. This demonstrates a fundamental flaw of the MUC F-measure: the metric intrinsically favors a system producing fewer entities and therefore lacks of discriminativity.

3.2.2 ACE-Value and CEAF

Now let us turn to ACE-value. Results in Table 4 are produced by a system trained on the ACE 2002 and 2004 training data and tested on a separate test set, which contains 853 reference entities. Both ACE-value and the mention-based CEAF penalizes systems over-producing or under-producing entities: ACE-value is maximum

Penalty	#sys-ent	ACE-value(%)	ϕ_3-CEAF
0.6	1221	88.5	0.726
0.4	1172	89.1	0.749
0.2	1145	89.4	0.755
0	1105	89.7	0.766
-0.2	1050	**89.7**	0.775
-0.4	1015	89.7	0.780
-0.6	990	89.5	0.782
-0.8	930	88.6	**0.794**
-1	891	86.9	0.780
-1.2	865	86.7	0.778
-1.4	834	85.6	0.769
-1.6	790	83.8	0.761

Table 4: Comparison of ACE-value and mention-based CEAF. The first column contains the penalty value in decreasing order. The second column contains the number of system-proposed entities. ACE-values are in percentage. The number of reference entities is 853.

when the penalty value is -0.2 and CEAF is maximum when the penalty value is -0.8. However, the optimal CEAF system produces 930 entities while the optimal ACE-value system produces 1050 entities. Judging from the number of entities, the optimal CEAF system is closer to the "truth" than the counterpart of ACE-value. This is not very surprising since ACE-value is a weighted metric while CEAF treats each mention and entity equally. As such, the two metrics have very weak correlation.

While we can make a statement such as "the system with penalty -0.8 puts about 79.4% mentions in right entities", it is hard to interpret the ACE-value numbers.

Another difference is that CEAF is symmetric[1], but ACE-Value is not. Symmetry is a desirable property. For example, when comparing inter-annotator agreement, a symmetric metric is independent of the order of two sets of input documents, while an asymmetric metric such as ACE-Value needs to state the input order along with the metric value.

4 Conclusions

A coreference performance metric – CEAF – is proposed in this paper. The CEAF metric is computed based on the best one-to-one map between reference entities and system entities. Finding the best one-to-one map is a maximum bipartite matching problem and can be solved by the Kuhn-Munkres algorithm. Two example entity-pair similarity measures (i.e., $\phi_3(\cdot,\cdot)$ and $\phi_4(\cdot,\cdot)$) are proposed, resulting one mention-based CEAF and one entity-based CEAF, respectively. It has been shown that the proposed CEAF metric has fixed problems associated with the MUC link-based F-measure and B-cube F-measure. The proposed metric also has better interpretability than ACE-value.

[1]This was pointed out by Nanda Kambhatla.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred.

The author would like to thank three reviewers and my colleagues, Hongyan Jing and Salim Roukos, for suggestions of improving the paper.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at The First International Conference on Language Resources and Evaluation (LREC'98)*, pages 563–566.

Egon Balas, Donald Miller, Joseph Pekny, and Paolo Toth. 1991. A parallel shortest augmenting path algorithm for the assignment problem. *Journal of the ACM (JACM)*, 38(4).

Francois Bourgeois and Jean-Claude Lassalle. 1971. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12).

R. Fletcher. 1987. *Practical Methods of Optimization*. John Wiley and Sons.

Anshul Gupta and Lexing Ying. 1999. Algorithms for finding maximum matchings in bipartite graphs. Technical Report RC 21576 (97320), IBM T.J. Watson Research Center, October.

H.W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(83).

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proc. of ACL*.

MUC-6. 1995. *Proceedings of the Sixth Message Understanding Conference(MUC-6)*, San Francisco, CA. Morgan Kaufmann.

MUC-7. 1998. *Proceedings of the Seventh Message Understanding Conference(MUC-7)*.

J. Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of SIAM*, 5:32–38.

NIST. 2003a. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

NIST. 2003b. Proceedings of ACE'03 workshop. Booklet, Alexandria, VA, September.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, , and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *In Proc. of MUC6*, pages 45–52.

Appendix: Kuhn-Munkres Algorithm

Let i index the reference entities $\mathcal{R}$ and j index the system entities $\mathcal{S}$, and $\phi(i,j)$ be the similarity between the i^{th} reference entity and the j^{th} system entity. Algebraically, the maximum bipartite matching can be stated as an integer programming problem:

$$\max_{\{x_{ij}\}} \phi(i,j)x_{ij} \quad (10)$$

$$\text{subject to:} \quad \sum_j x_{ij} \leq 1, \forall i \quad (11)$$

$$\sum_i x_{ij} \leq 1, \forall j \quad (12)$$

$$x_{ij} \in \{0,1\}, \forall i,j. \quad (13)$$

If $x_{ij} = 1$, the i^{th} reference entity and the j^{th} system entity are aligned. Constraint (11) (or (12)) implies that a reference (or system) entity cannot be aligned more than once with a system (or reference) entity.

Observe that the coefficients of (11) and (12) are unimodular. Thus, Constraint (13) can be replaced by

$$x_{ij} \geq 0, \forall i,j. \quad (14)$$

The dual (cf. pp. 219 of (Fletcher, 1987)) to the optimization problem (10) with constraints (11),(12) and (14) is:

$$\min_{\{u_i\},\{v_j\}} \sum_i u_i + \sum_j v_j \quad (15)$$

$$s.t.: \; u_i + v_j \geq \phi(i,j), \forall i,j \quad (16)$$

$$u_i \geq 0, \forall i \quad (17)$$

$$v_j \geq 0, \forall j. \quad (18)$$

The dual has the same optimal objective value as the primal.

It can be shown that the optimal conditions for the dual problem (and hence the maximum similarity match) are:

$$u_i + v_j = \phi(i,j), \text{if } (i,j) \text{ is aligned} \quad (19)$$

$$u_i = 0, \text{if } i \text{ is free (i.e., not aligned)} \quad (20)$$

$$v_j = 0, \text{if } j \text{ is free.} \quad (21)$$

The Kuhn-Munkres algorithm starts with an empty match and an initial feasible set of $\{u_i\}$ and $\{v_j\}$, and iteratively increases the cardinality of the match while satisfying the optimal conditions (19)-(21). Notice that conceptually, a matching problem with a rectangular matrix $[\phi(i,j)]$ can always reduce to a square one by padding zeros (this is not necessary in practice, see, for instance (Bourgeois and Lassalle, 1971)). For this reason, we state the Kuhn-Munkres algorithm for the case where $|\mathcal{R}| = |\mathcal{S}|$ (or $M = m$) in Algorithm 2. The proof of correctness is omitted due to space limit.

Note that $P_{aug}(i,j)$ on line 9 stands for the augmenting (i.e., a free node followed by an aligned node, followed by a free node, ...) path from i to j in the corresponding bipartite graph. $A \oplus P_{aug}(i,j)$ is understood as edge "exclusive-or:" if an edge (k,l) is in A and on the path $P_{aug}(i,j)$, it will be removed from A; if the edge is in either A or $P_{aug}(i,j)$, it will be added.

Algorithm 2 Kuhn-Munkres Algorithm

```
Input: similarity matrix: $[\phi(i,j)]$
Output: best match $A = \{(i,j)\}$ and similarity $\Phi$.
1:Initialize: $\forall i$, $u_i = \max_j \phi(i,j)$; $\forall j$, $v_j = 0$; $A = \emptyset$.
2:For $i = 1$ to $M$
3:  If $i$ is not free, Continue; EndIF.
4:  $X = \{i\}, Y = \emptyset$;
5:  While true
6:    $N(X) = \{l : \exists k \in X, s.t. \phi(k,l) = u_k + v_l\}$
7:    If $Y \subset N(X)$
8:      pick $j \in N(X) \setminus Y$
9:      If $j$ is free
10:       $A = A \oplus P_{aug}(i,j)$; break
11:     Else
12:       Find $i'$ such that $(i',j) \in A$.
13:       $X = X \cup \{i'\}, Y = Y \cup \{j\}$.
14:       Goto line 6.
15:     EndIf
16:   Else $Y == N(X)$
17:     $\delta = \min_{k \in X, l \in \bar{Y}} \{u_k + v_l - \phi(k,l)\}$
18:     $(\bar{i},\bar{j}) = arg\min_{k \in X, l \in \bar{Y}} \{u_k + v_l - \phi(k,l)\}$
19:     $u_k = u_k - \delta$ for $k \in X$.
20:     $v_l = v_l + \delta$ for $l \in Y$.
21:     $j = \bar{j}$. Goto line 9.
22:   EndIf
23:  EndWhile
24:EndFor
25:$\Phi = \sum_{(k,l) \in A} \phi(k,l)$.
26:Return $A$ and $\Phi$.
```

Improving Multilingual Summarization: Using Redundancy in the Input to Correct MT errors

Advaith Siddharthan and **Kathleen McKeown**
Columbia University Computer Science Department
1214 Amsterdam Avenue, New York, NY 10027, USA.
{as372,kathy}@cs.columbia.edu

Abstract

In this paper, we use the information redundancy in multilingual input to correct errors in machine translation and thus improve the quality of multilingual summaries. We consider the case of multi-document summarization, where the input documents are in Arabic, and the output summary is in English. Typically, information that makes it to a summary appears in many different lexical-syntactic forms in the input documents. Further, the use of multiple machine translation systems provides yet more redundancy, yielding different ways to realize that information in English. We demonstrate how errors in the machine translations of the input Arabic documents can be corrected by identifying and generating from such redundancy, focusing on noun phrases.

1 Introduction

Multilingual summarization is a relatively nascent research area which has, to date, been addressed through adaptation of existing extractive English document summarizers. Some systems (e.g. SUMMARIST (Hovy and Lin, 1999)) extract sentences from documents in a variety of languages, and translate the resulting summary. Other systems (e.g. Newsblaster (Blair-Goldensohn et al., 2004)) perform translation before sentence extraction. Readability is a major issue for these extractive systems. The output of machine translation software is usually errorful, especially so for language pairs such as Chinese or Arabic and English. The ungrammaticality and inappropriate word choices resulting from the use of MT systems leads to machine summaries that are difficult to read.

Multi-document summarization, however, has information available that was not available during the translation process and which can be used to improve summary quality. A multi-document summarizer is given a set of documents on the same event or topic. This set provides redundancy; for example, each document may refer to the same entity, sometimes in different ways. It is possible that by examining many translations of references to the same entity, a system can gather enough accurate information to improve the translated reference in the summary. Further, as a summary is short and serves as a surrogate for a large set of documents, it is worth investing more resources in its translation; readable summaries can help end users decide which documents they want to spend time deciphering.

Current extractive approaches to summarization are limited in the extent to which they address quality issues when the input is noisy. Some new systems attempt substituting sentences or clauses in the summary with similar text from extraneous but topic related English documents (Blair-Goldensohn et al., 2004). This improves readability, but can only be used in limited circumstances, in order to avoid substituting an English sentence that is not faithful to the original. Evans and McKeown (2005) consider the task of summarizing a mixed data set that contains both English and Arabic news reports. Their approach is to separately summarize information that is contained in only English reports, only Arabic reports, and in both. While the only-English and in-both information can be summarized by selecting text from English reports, the summaries of only-Arabic suffer from the same readability issues.

In this paper, we use principles from information theory (Shannon, 1948) to address the issue of readability in multilingual summarization. We take as input, multiple machine translations into English of a cluster of news reports in Arabic. This input is characterized by high levels of linguistic noise and by high levels of information redundancy (multiple documents on the same or related topics and multiple translations into English). Our aim is to use automatically acquired knowledge about the English language in conjunction with the information redundancy to perform error correction on the MT. The main benefit of our approach is to make machine summaries of errorful input easier to read and comprehend for end-users.

We focus on noun phrases in this paper. The amount of error correction possible depends on the amount of redundancy in the input and the depth of knowledge about English that we can utilize. We begin by tackling the problem of generating references to people in English summaries of Arabic texts (§2). This special case involves large amounts of redundancy and allows for relatively deep English language modeling, resulting in good error correction. We extend our approach to arbitrary NPs in §3.

The evaluation emphasis in multi-document summarization has been on evaluating content (not readability), using manual (Nenkova and Passonneau, 2004) as well as automatic (Lin and Hovy, 2003) methods. We evaluate readability of the generated noun phrases by computing precision, recall and f-measure of the generated version compared to multiple human models of the same reference, computing these metrics on n-grams. Our results show that our system performs significantly better on precision over two baselines (most frequent initial reference and randomly chosen initial reference). Precision is the most important of these measures as it is important to have a correct reference, even if we don't retain all of the words used in the human models.

2 References to people

2.1 Data

We used data from the DUC 2004 Multilingual summarization task. The Document Understanding Conference (http://duc.nist.gov) has been run annually since 2001 and is the biggest summarization evaluation effort, with participants from all over the world. In 2004, for the first time, there was a multilingual multi-document summarization task. There were 25 sets to be summarized. For each set consisting of 10 Arabic news reports, the participants were provided with 2 different machine translations into English (using translation software from ISI and IBM). The data provided under DUC includes 4 human summaries for each set for evaluation purposes; the human summarizers were provided a human translation into English of each of the Arabic New reports, and did not have to read the MT output that the machine summarizers took as input.

2.2 Task definition

An analysis of premodification in initial references to people in DUC human summaries for the monolingual task from 2001–2004 showed that 71% of premodifying words were either title or role words (eg. *Prime Minister, Physicist* or *Dr.*) or temporal role modifying adjectives such as *former* or *designate*. Country, state, location or organization names constituted 22% of premodifying words. All other kinds of premodifying words, such as *moderate* or *loyal* constitute only 7%. Thus, assuming the same pattern in human summaries for the multilingual task (cf. section 2.6 on evaluation), our task for each person referred to in a document set is to:

1. Collect all references to the person in both translations of each document in the set.
2. Identify the correct roles (including temporal modification) and affiliations for that person, filtering any noise.
3. Generate a reference using the above attributes and the person's name.

2.3 Automatic semantic tagging

As the task definition above suggests, our approach is to identify particular semantic attributes for a person, and generate a reference formally from this semantic input. Our analysis of human summaries tells us that the semantic attributes we need to identify are `role`, `organization`, `country`, `state`, `location` and `temporal modifier`. In addition, we also need to identify the `person name`. We used BBN's IDENTIFINDER (Bikel et al., 1999) to mark up person names, organizations and locations. We marked up countries and (American) states using a list obtained from the CIA factsheet[1].

[1]*http://www.cia.gov/cia/publications/factbook* provides a list of countries and states, abbreviations and adjectival forms, for example *United Kingdom/U.K./British/Briton* and *California/Ca./Californian*.

To mark up roles, we used a list derived from WordNet (Miller et al., 1993) hyponyms of the *person* synset. Our list has 2371 entries including multi-word expressions such as *chancellor of the exchequer*, *brother in law*, *senior vice president* etc. The list is quite comprehensive and includes roles from the fields of sports, politics, religion, military, business and many others. We also used WordNet to obtain a list of 58 temporal adjectives. WordNet classifies these as pre- (eg. *occasional*, *former*, *incoming* etc.) or post-nominal (eg. *elect*, *designate*, *emeritus* etc.). This information is used during generation. Further, we identified elementary noun phrases using the LT TTT noun chunker (Grover et al., 2000), and combined `NP of NP` sequences into one complex noun phrase. An example of the output of our semantic tagging module on a portion of machine translated text follows:

> ...<NP> <ROLE> representative </ROLE> of <COUNTRY> Iraq </COUNTRY> of the <ORG> United Nations </ORG> <PERSON> Nizar Hamdoon </PERSON> </NP> that <NP> thousands of people </NP> killed or wounded in <NP> the <TIME> next </TIME> few days four of the aerial bombardment of <COUNTRY> Iraq </COUNTRY> </NP>...

Our principle data structure for this experiment is the attribute value matrix (AVM). For example, we create the following AVM for the reference to Nizar Hamdoon in the tagged example above:

$$\begin{bmatrix} \texttt{name} & \text{Nizar Hamdoon} \\ \texttt{role} & \text{representative} \\ \texttt{country} & \text{Iraq } (arg1) \\ \texttt{organization} & \text{United Nations } (arg2) \end{bmatrix}$$

Note that we store the relative positions (*arg 1* and *arg 2*) of the country and organization attributes. This information is used both for error reduction and for generation as detailed below. We also replace adjectival country attributes with the country name, using the correspondence in the CIA factsheet.

2.4 Identifying redundancy and filtering noise

We perform coreference by comparing AVMs. Because of the noise present in MT (For example, words might be missing, or proper names might be spelled differently by different MT systems), simple name comparison is not sufficient. We form a coreference link between two AVMs if:

1. The last name and (if present) the first name match.
2. OR, if the role, country, organization and time attributes are the same.

The assumption is that in a document set to be summarized (which consists of related news reports), references to people with the same affiliation and role are likely to be references to the same person, even if the names do not match due to spelling errors. Thus we form one AVM for each person, by combining AVMs. For Nizar Hamdoon, to whom there is only one reference in the set (and thus two MT versions), we obtain the AVM:

$$\begin{bmatrix} \texttt{name} & \text{Nizar Hamdoon(2)} \\ \texttt{role} & \text{representative(2)} \\ \texttt{country} & \text{Iraq(2) } (arg1) \\ \texttt{organization} & \text{United Nations(2) } (arg2) \end{bmatrix}$$

where the numbers in brackets represents the counts of this value across all references. The *arg* values now represent the most frequent ordering of these organizations and countries in the input references. As an example of a combined AVM for a person with a lot of references, consider:

$$\begin{bmatrix} \texttt{name} & \text{Zeroual(24), Liamine Zeroual(20)} \\ \texttt{role} & \text{president(23), leader(2)} \\ \texttt{country} & \text{Algeria(18) } (arg1) \\ \texttt{organization} & \text{Renovation Party(2) } (arg1), \\ & \text{AFP(1) } (arg1) \\ \texttt{time} & \text{former(1)} \end{bmatrix}$$

This example displays common problems when generating a reference. Zeroual has two affiliations - Leader of the Renovation Party, and Algerian President. There is additional noise - the values *AFP* and *former* are most likely errors. As none of the organization or country values occur in the same reference, all are marked *arg1*; no relative ordering statistics are derivable from the input. For an example demonstrating noise in spelling, consider:

$$\begin{bmatrix} \texttt{name} & \text{Muammar Qaddafi(10),} \\ & \text{Muammar Gaddafi(10),} \\ & \text{Qaddafi(4), Gaddafi(4)} \\ \texttt{role} & \text{leader colonel(12), colonel(4)} \\ & \text{leader(3), minister(2), justice(1)} \\ \texttt{country} & \text{Libya(7) } (arg1) \\ \texttt{organization} & \text{Peace Country(2) } (arg2), \\ & \text{Country Peace(1) } (arg1) \end{bmatrix}$$

Our approach to removing noise is to:

1. Select the most frequent name with more than one word (this is the most likely full name).
2. Select the most frequent role.
3. Prune the AVM of values that occur with a frequency below an empirically determined threshold.

Thus we obtain the following AVMs for the three examples above:

$$\begin{bmatrix} \texttt{name} & \text{Nizar Hamdoon} \\ \texttt{role} & \text{representative} \\ \texttt{country} & \text{Iraq } (arg1) \\ \texttt{organization} & \text{United Nations } (arg2) \end{bmatrix}$$

$$\begin{bmatrix} \texttt{name} & \text{Liamine Zeroual} \\ \texttt{role} & \text{president} \\ \texttt{country} & \text{Algeria } (arg1) \end{bmatrix}$$

$$\begin{bmatrix} \texttt{name} & \text{Muammar Qaddafi} \\ \texttt{role} & \text{leader colonel} \\ \texttt{country} & \text{Libya } (arg1) \end{bmatrix}$$

This is the input semantics for our generation module described in the next section.

2.5 Generating references from AVMs

In order to generate a reference from the words in an AVM, we need knowledge about syntax. The syntactic frame of a reference to a person is determined by the role. Our approach is to automatically acquire these frames from a corpus of English text. We used the Reuters News corpus for extracting frames. We performed the semantic analysis of the corpus, as in §2.3; syntactic frames were extracted by identifying sequences involving locations, organizations, countries, roles and prepositions. An example of automatically acquired frames with their maximum likelihood probabilities for the role *ambassador* is:

ROLE=ambassador

(p=.35)	COUNTRY ambassador PERSON
(.18)	ambassador PERSON
(.12)	COUNTRY ORG ambassador PERSON
(.12)	COUNTRY ambassador to COUNTRY PERSON
(.06)	ORG ambassador PERSON
(.06)	COUNTRY ambassador to LOCATION PERSON
(.06)	COUNTRY ambassador to ORG PERSON
(.03)	COUNTRY ambassador in LOCATION PERSON
(.03)	ambassador to COUNTRY PERSON

These frames provide us with the required syntactic information to generate from, including word order and choice of preposition. We select the most probable frame that matches the semantic attributes in the AVM. We also use a default set of frames shown below for instances where no automatically acquired frames exist:

ROLE=<Default>

COUNTRY ROLE PERSON
ORG ROLE PERSON
COUNTRY ORG ROLE PERSON
ROLE PERSON

If no frame matches, organizations, countries and locations are dropped one by one in decreasing order of argument number, until a matching frame is found. After a frame is selected, any prenominal temporal adjectives in the AVM are inserted to the left of the frame, and any postnominal temporal adjectives are inserted to the immediate right of the role in the frame. Country names that are not objects of a preposition are replaced by their adjectival forms (using the correspondences in the CIA factsheet). For the AVMs above, our generation module produces the following referring expressions:

- Iraqi United Nations representative Nizar Hamdoon
- Algerian President Liamine Zeroual
- Libyan Leader Colonel Muammar Qaddafi

2.6 Evaluation

To evaluate the referring expressions generated by our program, we used the manual translation of each document provided by DUC. The drawback of using a summarization corpus is that only one human translation is provided for each document, while multiple model references are required for automatic evaluation. We created multiple model references by using the initial references to a person in the manual translation of each input document in the set in which that person was referenced. We calculated unigram, bigram, trigram and fourgram precision, recall and f-measure for our generated references evaluated against multiple models from the manual translations. To illustrate the scoring, consider evaluating a generated phrase *"a b d"* against three model references *"a b c d"*, *"a b c"* and *"b c d"*. The bigram precision is $1/2 = 0.5$ (one out of two bigrams in generated phrase occurs in the model set), bigram recall is $2/7 = 0.286$ (two out of 7 bigrams in the models occurs in the generated phrase) and f-measure ($f = 2p \times r/(p + r)$) is 0.364. For fourgrams, P, R and F are zero, as there is a fourgram in the models, but none in the generated NP.

We used 6 document sets from DUC'04 for development purposes and present the average P, R and F for the remaining 18 sets in Table 1. There were 210 generated references in the 18 testing sets. The table also shows the popular BLEU (Papineni et al., 2002) and NIST[2] MT metrics. We also provide two baselines - most frequent initial reference to the person in the input (Base1) and a randomly selected initial reference to the person (Base2). As Table 1 shows, Base1 performs better than random selection. This

[2]http://www.nist.gov/speech/tests/mt/resources/scoring.htm

UNIGRAMS	P_{av}	R_{av}	F_{av}
Generated	0.847*@	0.786	0.799*@
Base1	0.753*	0.805	0.746*
Base2	0.681	0.767	0.688

BIGRAMS	P_{av}	R_{av}	F_{av}
Generated	0.684*@	0.591	0.615*
Base1	0.598*	0.612	0.562*
Base2	0.492	0.550	0.475

TRIGRAMS	P_{av}	R_{av}	F_{av}
Generated	0.514*@	0.417	0.443*
Base1	0.424*	0.432	0.393*
Base2	0.338	0.359	0.315

FOURGRAMS	P_{av}	R_{av}	F_{av}
Generated	0.411*@	0.336	0.351*
Base1	0.320	0.360*	0.302
Base2	0.252	0.280	0.235

@ Significantly better than Base1
* Significantly better than Base2
(Significance tested using unpaired t-test at 95% confidence)

MT Metrics	Generated	Base1	Base2
BLEU	0.898	0.499	0.400
NIST	8.802	6.423	5.658

Table 1: Evaluation of generated reference

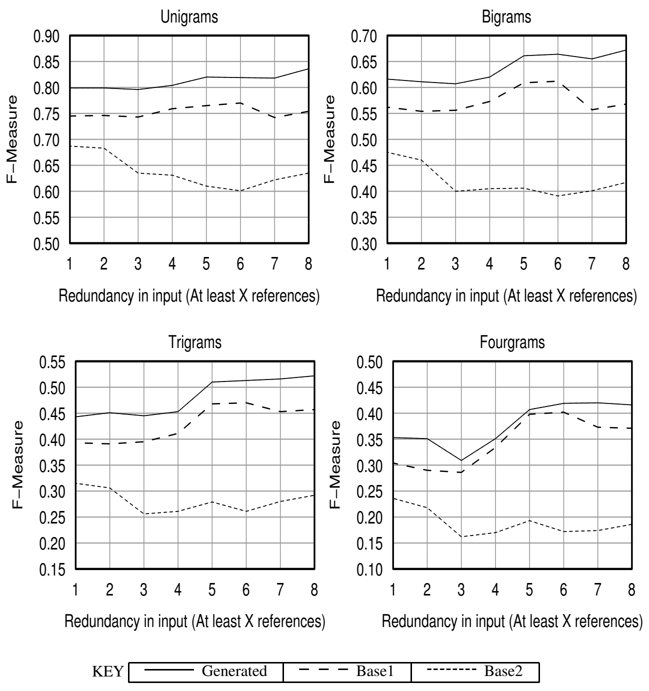


Figure 1: Improvement in F-measure for n-grams in output with increased redundancy in input.

is intuitive as it also uses redundancy to correct errors, at the level of phrases rather than words. The generation module outperforms both baselines, particularly on precision - which for unigrams gives an indication of the correctness of lexical choice, and for higher ngrams gives an indication of grammaticality. The unigram recall of 0.786 indicates that we are not losing too much information at the noise filtering stage. Note that we expect a low R_{av} for our approach, as we only generate particular attributes that are important for a summary. The important measure is P_{av}, on which we do well. This is also reflected in the high scores on BLEU and NIST.

It is instructive to see how these numbers vary as the amount of redundancy increases. Information theory tells us that information should be more recoverable with greater redundancy. Figure 1 plots f-measure against the minimum amount of redundancy. In other words, the value at X=3 gives the f-measure averaged over all people who were mentioned at least thrice in the input. Thus X=1 includes all examples and is the same as Table 1.

As the graphs show, the quality of the generated reference improves appreciably when there are at least 5 references to the person in the input. This is a convenient result for summarization because people who are mentioned more frequently in the input are more likely to be mentioned in the summary.

2.7 Advantages over using extraneous sources

Our approach performs noise reduction and generates a reference from information extracted from the machine translations. Information about a person can be obtained in other ways; for example, from a database, or by collecting references to the person from extraneous English-language reports. There are two drawbacks to using extraneous sources:

1. People usually have multiple possible roles and affiliations, so descriptions obtained from an external source might not be appropriate in the current context.
2. Selecting descriptions from external sources can change perspective — one country's terrorist is another country's freedom fighter.

In contrast, our approach generates references that are appropriate and reflect the perspectives expressed in the source.

3 Arbitrary noun phrases

In the previous section, we showed how accurate references to people can be generated using an information theoretic approach. While this is an important result in itself for multilingual summarization, the same approach can be extended to correct errors in noun phrases that do not refer to people. This extension is trickier to implement, however, because:

1. *Collecting redundancy*: Common noun coreference is a hard problem, even within a single clean English text, and harder still across multiple MT texts.

2. *Generating*: The semantics for an arbitrary noun phrase cannot be defined sufficiently for formal generation; hence our approach is to select the most plausible of the coreferring NPs according to an inferred language model. When sufficient redundancy exists, it is likely that there is at least one option that is superior to most.

Interestingly, the nature of multi-document summarization allows us to perform these two hard tasks. We follow the same theoretical framework (identify redundancy, and then generate from this), but the techniques we use are necessarily different.

3.1 Alignment of NPs across translations

We used the BLAST algorithm (Altschul et al., 1997) for aligning noun phrases between two translations of the same Arabic sentence. We obtained the best results when each translation was analyzed for noun chunks, and the alignment operation was performed over sequences of words and <*NP*> and <*/NP*> tags. BLAST is an efficient alignment algorithm that assumes that words in the two sentences are roughly in the same order from a global perspective. As neither of the MT systems used performs much clause or phrase reorganization, this assumption is not a problem for our task. An example of two aligned sentences is shown in figure 2. We then extract coreferring noun phrases by selecting the text between aligned <*NP*> and <*/NP*> tags; for example:

1. the Special Commission in charge of disarmament of Iraq's weapons of mass destruction
2. the Special Commission responsible disarmament Iraqi weapons of mass destruction

3.2 Alignment of NPs across documents

This task integrates well with the clustering approach to multi-document summarization (Barzilay, 2003), where sentences in the input documents are first clustered according to their similarity, and then one sentence is generated from each cluster. This clustering approach basically does at the level of sentences what we are attempting at the level of noun phrases. After clustering, all sentences within a cluster should represent similar information. Thus, similar noun phrases in sentences within a cluster are likely to refer to the same entities. We do noun phrase coreference by identifying lexically similar noun phrases within a cluster. We use *SimFinder* (Hatzivassiloglou et al., 1999) for sentence clustering and the f-measure for word overlap to compare noun phrases. We set a threshold for deciding coreference by experimenting on the 6 development sets (cf. §2.6)– the most accurate coreference occurred with a threshold of f=0.6 and a constraint that the two noun phrases must have at least 2 words in common that were neither determiners nor prepositions. For the reference to the *UN Special Commission* in figure 2, we obtained the following choices from alignments and coreference across translations and documents within a sentence cluster:

1. the United nations Special Commission in charge of disarmament of Iraq's weapons of mass destruction
2. the the United Nations Special Commission responsible disarmament Iraqi weapons of mass destruction
3. the Special Commission in charge of disarmament of Iraq's weapons of mass destruction
4. the Special Commission responsible disarmament Iraqi weapons of mass destruction
5. the United nations Special Commission in charge of disarmament of Iraq's weapons of mass destruction
6. the Special Commission of the United Nations responsible disarmament Iraqi weapons of mass destruction

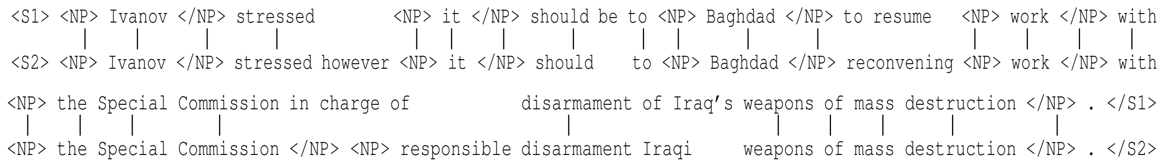
Larger sentence clusters represent information that is repeated more often across input documents; hence the size of a cluster is indicative of the importance of that information, and the summary is composed by considering each sentence cluster in decreasing order of size and generating one sentence from it. From our perspective of fixing errors in noun phrases, there is likely to be more redundancy in a large cluster; hence this approach is likely to work better within clusters that are important for generating the summary.

3.3 Generation of noun phrases

As mentioned earlier, formal generation from a set of coreferring noun phrases is impractical due to the unrestricted nature of the underlying semantics. We thus focus on selecting the best of the possible options — the option with the least garbled word order; for example, selecting 1) from the following:

1. the malicious campaigns in some Western media
2. the campaigns tendentious in some of the media Western European

The basic insight that we utilize is — when two words in a NP occur together in the original documents more often than they should by chance, it is likely they really should occur together in the generated NP. Our approach therefore consists of identifying collocations of length two. Let the number of words in the input documents be N. For each



```
<S1> <NP> Ivanov </NP> stressed         <NP> it </NP> should be to <NP> Baghdad </NP> to resume   <NP> work </NP> with
<S2> <NP> Ivanov </NP> stressed however <NP> it </NP> should    to <NP> Baghdad </NP> reconvening <NP> work </NP> with

<NP> the Special Commission in charge of           disarmament of Iraq's weapons of mass destruction </NP> . </S1>
<NP> the Special Commission </NP> <NP> responsible disarmament Iraqi     weapons of mass destruction </NP> . </S2>
```

Figure 2: Two noun chunked MT sentences (S1 and S2) with the words aligned using BLAST.

pair of words a and b, we use maximum likelihood to estimate the probabilities of observing the strings "$a\,b$", "a" and "b". The observed frequency of these strings in the corpus divided by the corpus size N gives the maximum likelihood probabilities of these events $p(a, b)$, $p(a)$ and $p(b)$. The natural way to determine how dependent the distributions of a and b are is to calculate their mutual information (Church and Hanks, 1991):

$$I(a, b) = \log_2 \frac{p(a, b)}{p(a) \times p(b)}$$

If the occurrences of a and b were completely independent of each other, we would expect the maximum likelihood probability $p(a, b)$ of the string "$a\ \ b$" to be $p(a) \times p(b)$. Thus mutual information is zero when a and b are independent, and positive otherwise. The greater the value of $I(a, b)$, the more likely that "$a\ \ b$" is a collocation. Returning to our problem of selecting the best NP from a set of coreferring NPs, we compute a score for each NP (consisting of the string of words $w_1...w_n$) by averaging the mutual information for each bigram:

$$Score(w_1...w_n) = \frac{\sum_{i=1}^{i=n-1} I(w_i, w_{i+1})}{n-1}$$

We then select the NP with the highest score. This model successfully selects *the malicious campaigns in some Western media* in the example above and *the United nations Special Commission in charge of disarmament of Iraq's weapons of mass destruction* in the example in §3.2.

3.4 Automatic Evaluation

Our approach to evaluation is similar to that for evaluating references to people. For each collection of coreferring NPs, we identified the corresponding model NPs from the manual translations of the input documents by using the BLAST algorithm for word alignment between the MT sentences and the corresponding manually translated sentence. Table 2 below gives the average unigram, bigram, trigram and fourgram precision, recall and f-measure for the

UNIGRAMS	P_{av}	R_{av}	F_{av}
Mutual information	0.615*@	0.658	0.607*
Base1	0.584	0.662	0.592
Base2	0.583	0.652	0.586
BIGRAMS	P_{av}	R_{av}	F_{av}
Mutual information	0.388*@	0.425*	0.374*@
Base1	0.340	0.402	0.339
Base2	0.339	0.387	0.330
TRIGRAMS	P_{av}	R_{av}	F_{av}
Mutual information	0.221*@	0.204*	0.196*@
Base1	0.177	0.184	0.166
Base2	0.181	0.171	0.160
FOURGRAMS	P_{av}	R_{av}	F_{av}
Mutual information	0.092*	0.090*	0.085*
Base1	0.078	0.080	0.072
Base2	0.065	0.066	0.061

@ Significantly better than Base1
* Significantly better than Base2
(Significance tested using unpaired t-test at 95% confidence)

MT Metrics	Mutual information	Base1	Base2
BLEU	0.276	0.206	0.184
NIST	5.886	4.979	4.680

Table 2: Evaluation of noun phrase selection

selected NPs, evaluated against the models. We excluded references to people as these were treated formally in §2. This left us with 961 noun phrases from the 18 test sets to evaluate. Table 2 also provides the BLEU and NIST MT evaluation scores.

We again provide two baselines - most frequent NP in the set (Base1) and a randomly selected NP from the set (Base2). The numbers in Table 2 are lower than those in Table 1. This is because generating references to people is a more restricted problem – there is less error in MT output, and a formal generation module is employed for error reduction. In the case of arbitrary NPs, we only select between the available options. However, the information theoretic approach gives significant improvement for the arbitrary NP case as well, particularly for precision, which is an indicator of grammaticality.

3.5 Manual Evaluation

To evaluate how much impact the rewrites have on summaries, we ran our summarizer on the 18 test sets, and manually evaluated the selected sentences and their rewritten versions for accuracy and fluency. There were 118 sentences, out of which 94 had at least one modification after the rewrite process. We selected 50 of these 94 sentences at random and asked 2 human judges to rate each sentence and its rewritten form on a scale of 1–5 for accuracy and fluency[3]. We used 4 human judges, each judging 25 sentence pairs. The original and rewritten sentences were presented in random order, so judges did not know which sentences were rewritten. Fluency judgments were made before seeing the human translated sentence, and accuracy judgments were made by comparing with the human translation. The average scores before and after rewrite were 2.08 and 2.26 respectively for fluency and 3.00 and 3.19 respectively for accuracy. Thus the rewrite operations increases both scores by around 0.2.

4 Conclusions and future work

We have demonstrated how the information redundancy in the multilingual multi-document summarization task can be used to reduce MT errors. We do not use any related English news reports for substituting text; hence our approach is not likely to change the perspectives expressed in the original Arabic news to those expressed in English news reports. Further, our approach does not perform any corrections specific to any particular MT system. Thus the techniques described in this paper will remain relevant even with future improvements in MT technology, and will be redundant only when MT is perfect. We have used the Arabic-English data from DUC'04 for this paper, but our approach is equally applicable to other language pairs. Further, our techniques integrate easily with the sentence clustering approach to multi-document summarization – sentence clustering allows us to reliably identify noun phrases that corefer across documents.

In this paper we have considered the case of noun phrases. In the future, we plan to consider other types of constituents, such as correcting errors in verb groups, and in the argument structure of verbs. This will result in a more generative and less extractive approach to summarization - indeed the case for generative approaches to summarization is more convincing when the input is noisy.

[3]We followed the DARPA/LDC guidelines from http://ldc.upenn.edu/Projects/TIDES/Translation/TranAssessSpec.pdf. For fluency, the scale was 5:Flawless, 4:Good, 3:Non-native, 2:Disfluent, 1:Incomprehensible. The accuracy scale for information covered (comparing with human translation) was 5:All, 4:Most, 3:Much, 2:Little, 1:None.

References

S.F. Altschul, T. L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 17(25):3389–3402.

R. Barzilay. 2003. *Information Fusion for Multidocument Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University, New York.

D. Bikel, R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231.

S. Blair-Goldensohn, D. Evans, V. Hatzivassiologlou, K. McKeown, A. Nenkova, R. Passonneau, B. Schiffman, A. Schlajiker, A. Siddharthan, and S. Siegelman. 2004. Columbia University at DUC 2004. In *Proceedings of DUC'04*, pages 23–30, Boston, USA.

K. Church and P. Hanks. 1991. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29.

D. Evans and K. McKeown. 2005. Identifying similarities and differences across english and arabic news. In *Proceedings of International Conference on Intelligence Analysis*, pages 23–30, McLean, VA.

C. Grover, C. Matheson, A. Mikheev, and M. Moens. 2000. LT TTT - A flexible tokenisation tool. In *Proceedings of LREC'00*, pages 1147–1154.

V. Hatzivassiloglou, J. Klavans, and E. Eskin. 1999. Detecting text similarity over short passages: exploring linguistic feature combinations via machine learning. In *Proceedings of EMNLP'99*, MD, USA.

E.H. Hovy and Chin-Yew Lin. 1999. Automated text summarization in summarist. In I. Mani and M. Maybury, editors, *Advances in Automated Text Summarization*, chapter 8. MIT Press.

C. Lin and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL'03*, Edmonton.

G.A. Miller, R. Beckwith, C.D. Fellbaum, D. Gross, and K. Miller. 1993. Five Papers on WordNet. Technical report, Princeton University, Princeton, N.J.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *HLT-NAACL 2004: Main Proceedings*, pages 145–152, Boston, MA, USA.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL'02*.

C. E. Shannon. 1948. A mathematical theory of communication. *Bell System Tech. Journal*, 27:379–423.

Error Detection Using Linguistic Features

Yongmei Shi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250
yshi1@umbc.edu

Lina Zhou
Information Systems Department
University of Maryland Baltimore County
Baltimore, MD 21250
zhoul@umbc.edu

Abstract

Recognition errors hinder the proliferation of speech recognition (SR) systems. Based on the observation that recognition errors may result in ungrammatical sentences, especially in dictation application where an acceptable level of accuracy of generated documents is indispensable, we propose to incorporate two kinds of linguistic features into error detection: lexical features of words, and syntactic features from a robust lexicalized parser. Transformation-based learning is chosen to predict recognition errors by integrating word confidence scores with linguistic features. The experimental results on a dictation data corpus show that linguistic features alone are not as useful as word confidence scores in detecting errors. However, linguistic features provide complementary information when combined with word confidence scores, which collectively reduce the classification error rate by 12.30% and improve the F measure by 53.62%.

1 Introduction

The proliferation of speech recognition (SR) systems is hampered by the ever-presence of recognition errors and the significant amount of effort involved in error correction. A user study (Sears et al., 2001) showed that users spent one-third of their time finding and locating errors and another one-third of the time correcting errors in a hand-free dictation task. Successfully detecting SR errors can speed up the entire process of error correction. Therefore, we focus on error detection in this study.

A common approach to detecting SR errors is annotating confidence at the word level. The majority of confidence annotation methods are based on feature combination, which follows two steps: (i) extract useful features characteristics of the correctness of words either from the inner components of an SR system (SR-dependent features) or from the recognition output (SR-independent features); and (ii) develop a binary classifier to separate words into two groups: correct recognitions and errors.

Various features extracted from different components of an SR system, such as the acoustic model, the language model, and the decoder, have been proven useful to detecting recognition errors (Chase, 1997; Pao et al., 1998; San-Segundo et al., 2001). Nonetheless, merely using these features is inadequate, because the information conveyed by these features has already been considered when SR systems generate the output. A common observation is that the combination of SR-dependent features can only marginally improve the performance achieved by using only the best single feature (Zhang and Rudnicky, 2001; Sarikaya et al., 2003). Hence information sources beyond the SR system are desired in error detection.

High-level linguistic knowledge is a good candidate for additional information sources. It can be extracted from the SR output via natural language processing, which compensates for the lack of high-level linguistic knowledge in a typical SR system. A user study (Brill et al., 1998) showed that humans can utilize linguistic knowledge at various levels to improve the SR output by selecting the best utterance hypotheses from N-best lists. Linguistic features from syntactic, semantic, and dialogue discourse analyses have proven their values in error detection in domain specific spoken dialogue systems, e.g. (Rayner et al., 1994; Carpenter et al., 2001; Sarikaya et al., 2003). However, few studies have investigated the merit of linguistic knowledge for error detection in dictation, a domain-independent application.

Transformation-based learning (TBL) is a rule-based learning method. It has been used in error correction (Mangu and Padmanabhan, 2001) and error detection (Skantze and Edlund, 2004). The rules learned by TBL show good interpretability as well as good performance. Although statistical learning methods have been widely used in confidence annotation (Carpenter et al., 2001; Pao et al., 1998; Chase, 1997), their results are difficult to interpret. Therefore, we select TBL to derive error patterns from the SR output in this study.

The rest of the paper is organized as follows. In Section 2, we review the extant work on utilizing linguistic features in error detection. In Section 3, we introduce linguistic features used in this study. In Section 4, we describe transformation-based learning and define the transformations, followed with reporting the experimental results in Section 5. Finally, we summarize the findings of this study and suggest directions for further research in Section 6.

2 Related Work

When the output of an SR system is processed, the entire utterance is available and thus utterance-level contextual information can be utilized. Features generated from high-level language processing such as syntactic and semantic analyses may complement the low-level language knowledge (usually n-gram) used in the SR systems.

Most of the previous work on utilizing linguistic features in error detection focused on utterance-level confidence measures. Most of features were extracted from the output of syntactic or semantic parsers, including full/robust/no parse, number of words parsed, gap number, slot number, grammar rule used, and so on (Rayner et al., 1994; Pao et al., 1998; Carpenter et al., 2001; San-Segundo et al., 2001). Some discourse-level features were also employed in spoken dialogue systems such as number of turns, and dialog state (Carpenter et al., 2001).

Several studies incorporated linguistic features into word-level confidence measures. Zhang and Rudnicky (2001) selected two features, i.e., parsing mode and slot backoff mode, extracted from the parsing result of Phoenix, a semantic parser. The above two features were combined with several SR-dependent features using SVM, which achieved a 7.6% relative classification error rate reduction over SR-dependent features on the data from CMU Communicator system.

Sarikaya et al. (2003) explored two sets of semantic features: one set from a statistical classer/parser, and the other set from a maximum entropy based semantic-structured language model. When combined with the posterior probability using the decision tree, both sets achieved about 13-14% absolute improvement on correct acceptance at 5% false acceptance over the baseline posterior probability on the data from IBM Communicator system.

Skantze and Edlund (2004) focused on lexical features (e.g., part-of-speech, syllables, and content words) and dialogue discourse features (e.g., previous dialogue act, and mentioned word), but did not consider parser-based features. They employed transformation-based learning and instance-based learning as classifiers. When combined with confidence scores, the linguistic features achieved 7.8% absolute improvement in classification accuracy over confidence scores on one of their dialogue corpora.

It is shown from the related work that linguistic features have merit in judging the correctness of words and/or utterances. However, such features have only been discussed in the context of conversational dialogue in specific domains such as ATIS (Rayner et al., 1994), JUPITER (Pao et al., 1998), and Communicator (Carpenter et al., 2001; San-Segundo et al., 2001; Zhang and Rudnicky, 2001; Sarikaya et al., 2003).

In an early study, we investigated the usefulness of linguistic features in detecting word errors in dictation recognition (Zhou et al., 2005). The linguistic features were extracted from the parsing result of the link grammar. The combination of linguistic features with various confidence score based features using SVM can improve F measure for error detection from 42.2% to 55.3%, and classification accuracy from 80.91% to 83.53%. However, parser-based features used were limited to the number of links that a word has.

3 Linguistic Features

For each output word, two sets of linguistic features are extracted: lexical features and syntactic features.

3.1 Lexical Features

For each word w, the following lexical features are extracted:

- word: w itself
- pos: part-of-speech tag from Brill's tagger (Brill, 1995)
- syllables: number of syllables in w, estimated based on the distribution patterns of vowels and consonants
- position: the position of w in the sentence: beginning, end, and middle

3.2 Syntactic Features

Speech recognition errors may result in ungrammatical sentences under the assumption that the speaker follows grammar rules while speaking. Such an assumption holds true especially for dictation application because the general purpose of dictation is to create understandable documents for communication.

Syntactic parsers are considered as the closest approximation to this intuition since there is still a lack of semantic parsers for the general domain. Moreover, robust parsers are preferred so that an error in a recognized sentence does not lead to failure in parsing the entire sentence. Furthermore, lexicalized parsers are desired to support error detection at the word level. As a result, we select *Link Grammar*[1] to generate syntactic features.

[1]Available via http://www.link.cs.cmu.edu/link/

3.2.1 Link Grammar

Link Grammar is a context-free lexicalized grammar without explicit constituents (Sleator and Temperley, 1993). In link grammar, rules are expressed as link requirements associated with words. A link requirement is a set of disjuncts, each of which represents a possible usage of the word. A sequence of words belongs to the grammar if the result linkage is a planar, connected graph in which at most one link is between each word pair and no cross link exists. Link grammar supports robust parsing by incorporating null links (Grinberg et al., 1995).

3.2.2 Features from Link Grammar

We hypothesize that a word without any link in a linkage of the sentence is a good indicator of the occurrence of errors. Either the word itself or words around it are likely to be erroneous. It has been shown that null links can successfully ignore false starts and connect grammatical phrases in ungrammatical utterances, which are randomly selected from the Switchboard corpus (Grinberg et al., 1995).

A word with links may still be an error, and its correctness may affect the correctness of words linked to it, especially those words connected with the shortest links that indicate the closest connections.

Accordingly, for each word w, the following features are extracted from the parsing result:

- haslink: whether w has left links, right links, or no link
- llinkto/rlinkto: the word to which w links via the shortest left/right link

An example of parsing results is illustrated in Figure 1. Links are represented with dotted lines which are annotated with labels (e.g., Wd, Xp) representing link types. In Figure 1, word "since" has no link, and word "around" has one left link and one right link. The word that has the shortest left link to "world" is "the".

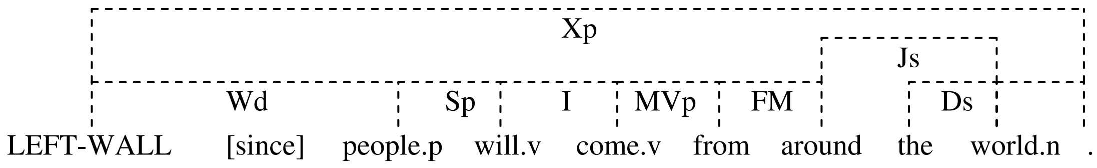


Figure 1: An Example of Parsing Results of Link Grammar

4 Error Detection based on Transformation-Based Learning

4.1 Transformation-Based Learning

Transformation-Based Learning is a rule-based approach, in which rules are automatically learned from the data corpus. It has been successfully used in many natural language applications such as part-of-speech tagging (Brill, 1995). Three prerequisites for using TBL are: an initial state annotator, a set of possible transformations, and an objective function for choosing the best transformations.

Before learning, the initial state annotator adds labels to the training data. The learning goes through the following steps iteratively until no improvement can be achieved: (i) try each possible transformation on the training data, (ii) score each transformation with the objective function and choose the one with the highest score, and (iii) apply the selected transformation to update the training data and append it to the learned transformation list.

4.2 Error Detection Based on TBL

Pre-defined transformation templates are the rules allowed to be used, which play a vital role in TBL. The transformation templates are defined in the following format:

Change the word label of a word w from X to Y, if condition C is satisfied

where, X and Y take binary values: 1 (correct recognition) and -1 (error). Each condition C is the conjunction of sub-conditions in form of $f \ op \ v$, where f represents a feature, v is a possible categorical value of f, and op is the possible operations such as $<$, $>$ and $=$.

In addition to the linguistic features introduced in Section 3, two other features are used:

- word confidence score (CS): an SR dependent feature generated by an SR system.
- word label (label): the target of the transformation rules. Using it as a feature enables the propagation of the effect of preceding rules.

As shown in Table 1, conditions are classified into three categories based on the incrementally enlarged context from which features are extracted: *word alone*, *local context*, and *sentence context*. The three categories are further split into seven groups according to the features they used.

- **L**: the correctness of w depends solely on itself. Conditions only include lexical features of w.
- **Local**: the correctness of w depends not only on itself but also on its surrounding words. Conditions incorporate lexical features of surrounding words as well as those of w. Furthermore, word labels of surrounding words are also employed as a feature to capture the effect of the correctness of surrounding words of w.
- **Long**: the scope of conditions for the correctness of w is expanded to include syntactic features. Syntactic features of w and its surrounding words as well as the features in Local are incorporated into conditions. In addition, the lexical features and word labels of words that have the shortest links to w are also incorporated.
- **CS**: the group in which conditions only include confidence scores of w.
- **LCS, CSLocal, CSLong**: these three groups are generated by combining the features from L, Local, and Long with the confidence scores of w as an additional feature respectively.

$lrHaslink$ and $llinkLabel$ are combinations of basic features. $lrHaslink$ represents whether the preceding word and the following word have links,

Category	Group	Example
Word Alone	CS	$cs(w_i) < c_i$
	L	$position(w_i) = t_i$ & $syllables(w_i) = s_i$
	LCS	$cs(w_i) < c_i$ & $pos(w_i) = p_i$
Local Context	Local	$position(w_i) = t_i$ & $label(w_{i-1}) = l_{i-1}$ & $word(w_i) = d_i$
	CSLocal	$cs(w_i) < c_i$ & $position(w_i) = t_i$ & $label(w_{i-1}) = l_{i-1}$ & $label(w_{i+1}) = l_{i+1}$
Sentence Context	Long	$position(w_i) = t_i$ & $lrHaslink(w_i) = h_i$ & $haslink(w_i) = hl_i$
	CSLong	$cs(w_i) < c_i$ & $position(w_i) = t_i$ & $llinkLabel(w_i) = ll_i$ & $pos(w_i) = p_i$

Table 1: Condition Categories and Examples

and $llinkLabel$ represents the label of the word to which w has the shortest left link. c_i, t_i, s_i, p_i, l_i, d_i, h_i, hl_i, and ll_i are possible values of the corresponding features.

The initial state annotator initializes all the words as correct words. A Prolog based TBL tool, μ-TBL (Lager, 1999) [2] is used in this study. Classification accuracy is adopted as the objective function. For each transformation, its *positive effect* (*PE*) is the number of words whose labels are correctly updated by applying it, and its *negative effect* (*NE*) is the number of words wrongly updated. Two cut-off thresholds are used to select transformations with strong positive effects: net positive effect ($PE - NE$), and the ratio of positive effect ($PE/(PE + NE)$).

5 Experimental Results and Discussion

Experiments were conducted at several levels. Starting with transformation rules with word alone conditions, additional rules with local context and sentence context conditions were incorporated incrementally by enlarging the scope of the context. As such, the results help us not only identify the additional contribution of each condition group to the task of error detection but also reveal the importance of enriching contextual information to error detection.

5.1 Data Corpus

The data corpus was collected from a user study on a composition dictation task (Feng et al., 2003). A total of 12 participants were native speakers and none of them used their voice for professional purposes. Participants spoke to IBM ViaVoice (Millennium edition), which contains a general vocabulary of 64,000 words. The dictation task was completed in a quiet lab environment with high quality microphones.

During the study, participants were given one predesigned topic and instructed to compose a document of around 400 words on that topic. Before starting the dictation, they completed enrollments to build personal profiles and received training on finishing the task with a different topic. They were asked to make corrections only after they finished composing a certain length of text. The data corpus consists of the recognition output of their dictations excluding corrections. Word recognition errors were first marked by the participants themselves and then validated by researchers via cross-referencing the recorded audios. The data corpus contains 4,804 words.

5.2 Evaluation Metrics

To evaluate the overall performance of the error detection, classification error rate (CER) (Equation 1), commonly used metric to evaluate classifiers, is used. CER is the percentage of words that are wrongly classified.

$$CER = \frac{\#\ of\ wrongly\ classified\ words}{total\#\ of\ words} \quad (1)$$

The baseline CER is derived by assuming all the words are correct, and it has the value as the ratio of the total number of insertion and substitution errors to the total number of output words.

Precision (PRE) and recall (REC) on errors are used to measure the performance of identifying er-

[2] Available via http://www.ling.gu.se/˜lager/mutbl.html

rors. PRE is the percentage of words classified as errors that are in fact recognition errors. REC denotes the proportion of actual recognition errors that are categorized as errors by the classifier. In addition, F measure (Equation 2), a single-valued metric reflecting the trade-off between PRE and REC, is also used. The baselines of PRE, REC, and F for error are zeros, for all of the output words are assumed correct.

$$F = \frac{2 * PRE * REC}{PRE + REC} \quad (2)$$

5.3 Results

3-fold cross-validation was used to test the system. When dividing the data corpus, sentence is treated as an atomic unit. The 3-fold cross-validation was run 9 times, and the average performance is reported in Table 2. The labels of rule combinations are defined by the connections of several symbols defined in Section 4.2. For each rule combination, the types of rules can be included are decided by all the possible combinations of those symbols which are in Table 1. For example, L-CS-Local-Long includes rules with conditions L, CS, Local, Long, LCS, CSLocal and CSLong.

The threshold of net positive effect is set to 5 to ensure that enough evidence has been observed, and that of the ratio of the positive effect is set to 0.5 to ensure that selected transformations have the positive effects.

For the combinations without CS, L-Local-Long achieves the best performance in terms of both CER and F measure. A relative improvement of 4.85% is achieved over the baseline CER, which is relatively small. One possible explanation concerns the large vocabulary size in the data set. Although the participants were asked to compose the documents on the same topic, the word usage was greatly diversified. An analysis of the data corpus shows that the vocabulary size is 993.

Despite its best performance in linguistic feature groups, L-Local-Long produces worse performance than CS in both CER and F measure. Therefore, linguistic features by themselves are not as useful as confidence scores.

When linguistic features are combined with CS, they provide additional improvement. L-CS achieves a 4.58% relative improvement on CER and a 31.37% relative improvement on F measure over CS. L-CS-Local only achieves marginal improvement on CER and a 7.54% relative improvement on F measure over L-CS.

The best performance is generated by L-CS-Local-Long. In particular, it boosts CER by a relative improvement of 12.30% over CS and a relative improvement of 7.02% over L-CS-Local. In addition, it improves F measure by 53.62% and 8.74% in comparison with CS and L-CS-Local respectively. Therefore, enlarging the scope of context can lead to improved performance on error detection.

It is revealed from Table 2 that the improvement on F measure is due to the improvement on recall without hurting the precision. After combining linguistic features with CS, L-CS and L-CS-Local-Long achieve 43.77% and 75.57% relative improvements on recall over CS separately. Hence, the linguistic features can improve the system's ability in finding more errors. Additionally, L-CS-Local-Long achieves a 7.32% relative improvement on precision over CS.

The average numbers of learned rules are shown in Table 2. With the increased number of possible used pre-defined rules, the number of learned rules increases moderately. L-CS-Local-Long and L-CS-Local have the largest number of rules, 14, which is rather a small set of rules. As discussed above, these rules are straightforward and easy to understand.

Figure 2 shows CERs when the learned rules are incrementally applied in one run for L-CS-Local-Long. Three lines represent each of the three folds separately, and the number of learned rules differs among folds.

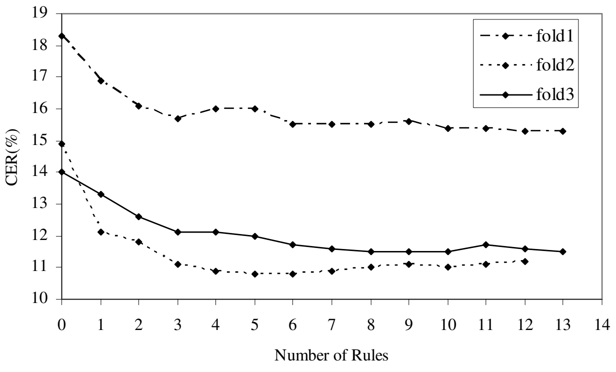


Figure 2: Relations of CERs with Number of Rules

Combination	Mean CER (%)	Std. Dev	Mean PRE (%)	Mean REC (%)	Mean F (%)	Mean # of rules
Baseline	15.66	0.06	-	-	-	-
L	15.55	0.11	61.85	2.04	3.88	3
L-Local	15.58	0.14	60.88	2.19	4.17	4
L-Local-Long	14.90	0.10	61.67	13.83	22.37	8
CS	14.64	0.09	61.03	21.98	31.50	1
L-CS	13.97	0.15	61.48	31.60	41.38	8
L-CS-Local	13.81	0.18	61.28	35.52	44.50	14
L-CS-Local-Long	12.84	0.21	65.50	38.59	48.39	14

Table 2: Performance of Transformation Rule Combinations

After the first several rules are applied, CERs drop significantly. Then the changes in CERs become marginal as additional rules are applied. The fold 1 and 3 reach the lowest CER after the last rule is applied, and fold 2 reaches the lowest CERs in the middle. Thus, the top ranked rules are mostly useful.

One advantage of TBL is that the learning result can be easily interpreted. The following is the top six rules learned in fold 3 in Figure 2.

Mark a word as an error, if :

- its confidence score is less than 0; it is in the middle of a sentence; and it is a null-link word.
- its confidence score is less than -5; it is in the middle of a sentence; and it has links to preceding words.
- its confidence score is less than 0; it is the first word of a sentence; and it is a null-link word.
- its confidence score is less than 2; it is in the middle of a sentence; it has 1 syllable; and the word following it also has 1 syllable and is an error.
- its confidence score is less than -1; and both its preceding and following words are errors.

Mark a word as a correct word, if :

- its confidence score is greater than -1; and both its preceding and following words are correct words.

All of the above six rules include word confidence score as a feature. Rule 1 and rule 3 suggest that null-link words are good indicators of errors, which confirms our hypothesis. Rule 2 shows that a word with low confidence score may also be an error even if it is part of the linkage of the sentence. Rule 4 shows continuous short words are possible errors. Rule 5 indicates that a word with low confidence score may be an error if its surrounding words are errors. Rule 6 is a rule to compensate for the wrongly labeled words by previous rules.

6 Conclusion and Future Works

We introduced an error detection method based on feature combinations. Transformation-based learning was used as the classifier to combine linguistic features with word confidence scores. Two kinds of linguistic features were selected: lexical features extracted from words themselves, and syntactic features from the parsing result of link grammar. Transformation templates were defined by varying scope of the context. Experimental results on a dictation corpus showed that although linguistic features alone were not as useful as word confidence scores to error detection, they provided complementary information when combined with word confidence score. Moreover, the performance of error detection was improved incrementally as the scope of context was enlarged, and the best performance was achieved when sentence context was considered. In particular, enlarging the context modeled by linguistic features improved the capability of error detection by finding more errors without deteriorating and even improving the precision.

The proposed method has been tested using a dictation corpus on a topic related to office environment. We are working on evaluating the method on spontaneous dictation utterances from the CSR-II corpus, and other monologue corpora such as Broadcast News. The method can be extended by incorporating lexical semantic features from the semantic analysis of recognition output to detect semantic errors that are likely overlooked by syntactic analysis.

Acknowledgement

This work is supported by the National Science Foundation under Grant# 0328391. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).

References

Eric Brill, Radu Florian, John C. Henderson, and Lidia Mangu. 1998. Beyond n-grams: Can linguistic sophistication improve language modeling? In *Proceedings of COLING/ACL*, pages 186–190.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

Paul Carpenter, Chun Jin, Daniel Wilson, Rong Zhang, Dan Bohus, and Alex Rudnicky. 2001. Is this conversation on track? In *Proceedings of Eurospeech*, pages 2121–2124.

Lin L. Chase. 1997. *Error-Responsive Feedback Mechanisms for Speech Recognizers*. Ph.D. thesis, School of Computer Science, CMU, April.

Jinjuan Feng, Andrew Sears, and Clare-Marie Karat. 2003. A longitudinal investigation of hands-free speech based navigation during dictation. Technical report, UMBC.

Dennis Grinberg, John Lafferty, and Daniel Sleator. 1995. A robust parsing algorithm for link grammars. Technical Report CMU-CS-95-125, Carnegie Mellon University.

Torbjörn Lager. 1999. The μ-tbl system: Logic programming tools for transformation-based learning. In *Proceedings of the third international workshop on computational natural language learning*.

Lidia Mangu and Mukund Padmanabhan. 2001. Error corrective mechanisms for speech recognition. In *Proceedings of ICASSP*, volume 1, pages 29–32.

Christine Pao, Philipp Schmid, and James Glass. 1998. Confidence scoring for speech understanding systems. In *Proceedings of ICSLP*, pages 815–818.

Manny Rayner, David Carter, Vassilios Digalakis, and Patti Price. 1994. Combining knowledge sources to reorder n-best speech hypothesis lists. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 212–217.

Rubén San-Segundo, Bryan Pellom, Kadri Hacioglu, and Wayne Ward. 2001. Confidence measures for spoken dialogue systems. In *Proceedings of ICASSP*, volume 1, pages 393–396.

Ruhi Sarikaya, Yuqing Gao, and Michael Picheny. 2003. Word level confidence measurement using semantic features. In *Proceedings of ICASSP*, volume 1, pages 604–607.

Andrew Sears, Clare-Marie Karat, Kwesi Oseitutu, Azfar S. Karimullah, and Jinjuan Feng. 2001. Productivity, satisfaction, and interaction strategies of individuals with spinal cord injuries and traditional users interacting with speech recognition software. *Universal Access in the Information Society*, 1(1):4–15, June.

Gabriel Skantze and Jens Edlund. 2004. Early error detection on word level. In *Proceedings of Robust*.

Daniel Sleator and Davy Temperley. 1993. Parsing english with a link grammar. In *Proceedings of the third international workshop on parsing technologies*.

Rong Zhang and Alexander I. Rudnicky. 2001. Word level confidence annotation using combinations of features. In *Proceedings of Eurospeech*, pages 2105–2108.

Lina Zhou, Yongmei Shi, Jinjuan Feng, and Andrew Sears. 2005. Data mining for detecting errors in dictation speech recognition. *IEEE Transactions on Speech and Audio Processing, Special Issues on Data Mining of Speech, Audio and Dialog*, 13(5), September.

Semantic Similarity for Detecting Recognition Errors in Automatic Speech Transcripts

Diana Inkpen
School of Information Technology and Engineering
University of Ottawa
Ottawa, ON, K1N 6H5, Canada
diana@site.uottawa.ca

Alain Désilets
Institute for Information Technology
National Research Council of Canada
Ottawa, ON, K1AOR6, Canada
alain.desilets@nrc-cnrc.gc.ca

Abstract

Browsing through large volumes of spoken audio is known to be a challenging task for end users. One way to alleviate this problem is to allow users to gist a spoken audio document by glancing over a transcript generated through Automatic Speech Recognition. Unfortunately, such transcripts typically contain many recognition errors which are highly distracting and make gisting more difficult. In this paper we present an approach that detects recognition errors by identifying words which are semantic outliers with respect to other words in the transcript. We describe several variants of this approach. We investigate a wide range of evaluation measures and we show that we can significantly reduce the number of errors in content words, with the trade-off of losing some good content words.

1 Introduction

Spoken audio documents are becoming more and more common place due to the rising popularity of technologies such as: video and audio conferencing, video web-casting and digital cameras for the consumer market. Unfortunately, speech documents are inherently hard to browse because of their transient nature. For example, imagine trying to locate the audio segment in the recording of a 60-minute meeting, where John talked about project X. Typically, this would require fast forwarding through the audio by some amount, then listening and trying to remember if the current segment was spoken before or after the desired segment, then fast-forwarding or backtracking by a small amount, and so on.

One way to make audio browsing of audio documents more efficient is to allow the user to navigate through a textual transcript that is cross-referenced with corresponding time points into the original audio (Nakatani *et al.* 1998; Hirschberg *et al.* 1999). Such transcripts can easily be produced with Automatic Speech Recognition (ASR) systems today. Unfortunately, such transcripts typically contain recognition errors that make them hard to browse and understand. Although Word Error Rates (WER) of the order of 20% can be achieved for broadcast quality audio, the WER for more common situations (ex: less-than-broadcast quality recordings of meetings) is typically in the order of 50% or more.

The work we present in this paper aims at automatically identifying recognition errors and removing them from the transcript, in order to make gisting and browsing of the corresponding audio more efficient. For example, consider the following portion of a transcript that was produced with the Dragon NaturallySpeaking speech recognition system from the audio of a meeting:

"Weenie to decide quickly whether local for large expensive plasma screen aura for a bunch of smaller and cheaper ones and Holland together"

Now consider the following **filtered transcript** where recognition errors were automatically blotted out using our proposed algorithm*:*

" ... to decide quickly whether ... large expensive plasma screen ... for a bunch of smaller and cheaper ones and ... together"

We believe that transcripts like this second one may be more efficient for gisting and browsing the

content of the original audio whose **correct transcript** is:
"We need to decide quickly whether we will go for a large expensive plasma screen or for a bunch of smaller and cheaper ones and tile them together."

Our approach to filtering recognition errors is to identify **semantic outliers**. By this, we mean words that do not cohere well semantically with other words in the transcript. More often than not, such outliers turn out to be mistranscribed words. We present several variants of an algorithm for identifying semantic outliers, and evaluate them in terms of how well they are able to filter out recognition errors.

2 Related Work

Hirschberg *et al.* (1999), and Nakatani *et al.* (1998) proposed the idea of using automatic transcripts for gisting and navigating audio documents. Text-based summarization techniques on automatic speech transcription have also been used. For example, the method of Désilets *et al.* (2001) was found to produce accurate keyphrases for transcriptions with Word Error Rates (WER) in the order of 25%, but performance was less than ideal for transcripts with WER in the order of 60%. With such transcripts, a large proportion of the extracted keyphrases included serious transcription errors. Inkpen and Désilets (2004) presented an experiment that filters out errors in keywords extracted from speech, by identifying the keywords that are not semantically close to the rest of the keywords.

Semantic similarity measures were used for many tasks. Two examples are: real-word error correction (Budanitsky and Hirst, 2000) and answering synonym questions (Turney, 2001), (Jarmasz and Szpakowicz, 2003).

There is a lot of research on confidence measures for identifying errors in speech recognition output. Most papers on this topic use information that is internal to the ASR system, generated by the decoder during the recognition process. Examples are likelihood ratios derived by a Viterbi decoder (Gillick *et al.*, 1997), measures of competing words at a word boundary (Cox and Rose, 1996), word score densities in N-best lists, and various acoustic and phonetic features. Machine learning techniques were used to identify the best combinations of features for classification (Chase, 1997) (Schaaf and Kemp, 1997) (Ma *et al.*, 2001) (Skantze and Edlund, 2004) (Zhou and Meng, 2004) (Zhou *et al.*, 2005). Some of these methods achieve good performance, although they use different test sets and report different evaluation measures from the set we enumerate in Section 6.

In our work, we use information that is external to the ASR system, because new knowledge seems likely to help in the detection of semantic outliers. In this respect, the work of Cox and Dasmahapatra (2000) is closest to ours. They compared the accuracy of a measure based on Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997) to an ASR-based confidence measure, and found that the ASR-based measure (using N-best lists) outperformed the LSA approach. While the N-best lists approach was better at the high-Recall end of the spectrum, the LSA was better at the high-Precision end. They also showed that a hybrid combination of the two approaches worked best. Our work is similar to the LSA-based part of Cox and Dasmahapatra, except that we use **Point-wise Mutual Information** (PMI) instead of LSA. Because PMI scales up to very large corpora, it has been shown to work better than LSA for assessing the semantic similarity of words (Turney, 2001). Another distinguishing feature is that Cox and Dasmahapatra only looked at transcripts with moderate WER, whereas we additionally evaluate the technique for the purpose of doing error filtering on transcripts with high WER, which are more typical of non-broadcast conversational audio.

3 The Data

We evaluated our algorithms on a randomly selected subset of 100 stories from the TDT2 English Audio corpus. We conducted experiments with two types of automatically-generated speech transcripts. The first ones were generated by the NIST/BBN time-adaptive speech recognizer and have a moderate WER (27.6%), which is representative of what can be obtained with a speaker-independent ASR system tuned for the Broadcast News domain. In the rest of this paper, we refer to these moderate accuracy transcripts as the **BBN dataset**. The second set of transcripts was obtained using the Dragon NaturallySpeaking speaker-dependent recognizer. Their WER (62.3%) was much higher because the voice model was not trained for speaker-independent broadcast quality audio. These transcripts approximate the type of high WER seen in more casual less-than-broadcast quality audio. We refer to these transcripts as the **Dragon dataset**.

4 The method

Our algorithm tries to detect recognition errors by identifying and filtering semantic outliers in the transcripts. In other words, it declares as recognition errors all the words with low semantic similarity to other words in the transcript. The algorithm focuses on **content words**, i.e., words that do not appear in a list of 779 stopwords (including closed-class words, such as prepositions, articles, etc.). The reason to ignore stopwords is that they tend to co-occur with most words, and are therefore semantically coherent with most words. The basic algorithm for determining if a word **w** is a recognition error is as follows.

1. Compute the **neighborhood** N(w) of w as the set of content words that occur before and after w in a context window (including w itself).

2. Compute **pair-wise semantic similarity** scores $S(w_i, w_j)$ between all pairs of words $w_i \neq w_j$ (including w) in the neighborhood N(w), using a semantic similarity measure. Scale up those $S(w_i, w_j)$ by a constant so that they are all non-negative, and the smallest one is 0.

3. For each w_i in the neighborhood N(w) (including w), compute its **semantic coherence** $SC(w_i)$. by "aggregating" the pair-wise semantic similarities $S(w_i, w_j)$ of w_i with all its neighbors ($w_i \neq w_j$) into a single number.

4. Let SC_{avg} be the average of $SC(w_i)$ over all w_i in the neighborhood N(w).

5. Label w as a recognition error if $SC(w) < K \cdot SC_{avg}$, where K is a parameter that allows us to control the amount of error filtering (K% of the average semantic coherence score). Low values of K mean little error filtering and high values of K mean a lot of error filtering.

We tested a number of variants of Steps 1-3. For Step 1, we experimented with two ways of computing the neighborhood N(w). The first approach was to set N(w) to be all the words in the transcript (the **All** variant). The second neighborhood approach was to set N(w) to be the set of 10 content words before and after w in the transcript (the **Window** variant).

For Step 2 we experimented with two different measures for evaluating the pair-wise semantic similarities $S(w_i, w_j)$. The first measure used a hand-crafted dictionary (the **Roget** variant) whereas the second one used a statistical measure based on a large corpus (the **PMI** variant).

For Step 3 we experimented with different schemes for "aggregating" the pair-wise semantic similarities $S(w_i, w_j)$ into a single semantic coherence number $SC(w_i)$ for a given word w_i. The first aggregation scheme was simply to average the $SC(w_i)$ values (the **AVG** variant). Note that with this scheme, we filter words that do not cohere well with all the words in the neighborhood N(w). This might be too aggressive in the case of the **All** variant, especially for longer or multi-topic audio documents. Therefore, we investigated other aggregation schemes that only required words to cohere well with a subset of the words in N(w). The second aggregation scheme was to set $SC(w_i)$ to the value of the most similar neighbor in N(w) (the **MAX** variant). The third aggregation scheme was to set $SC(w_i)$ to the average of the 3 most similar neighbors in N(w) (the **3MAX** variant).

Thus, there are altogether 2x2x3 = 12 possible configurations of the algorithm. In the rest of this paper, we will refer to specific configurations using the following naming scheme: **Step1Variant-Step2Variant-Step3Variant**. For example, All-PMI-AVG means the configuration that uses the All variant of Step 1, the PMI variant of Step 2, and the AVG variant of step 3.

It is worth noting that all configurations of this algorithm are computationally intensive, mainly because of Step 2. However, since our aim is to provide transcripts for browsing audio recordings, we do not have to correct errors in real time.

5 Choosing a semantic similarity measure

Semantic similarity refers to the degree with which two words (two concepts) are related. For example, most human judges would agree that *paper* and *pencil* are more closely related than *car* and *toothbrush*. We use the term *semantic similarity* in this paper in a more general sense of *semantic relatedness* (two concepts can be related by their context of use without necessarily being similar).

There are three types of semantic similarity measures: dictionary-based (lexical taxonomy structure), corpus-based, and hybrid. Most of the dictionary-based measures use path length in WordNet – for example (Leacock and Chodorow, 1998), (Hirst and St-Onge, 1998). The corpus-based measures use some form of vector similarity. The cosine measure uses frequency counts in its vectors and cosine to compute similarity; the simpler methods use binary vectors and compute coefficients such as: Matching, Dice, Jaccard, and Overlap. Examples of hybrid measures, based on WordNet and small corpora, are: Resnik (1995), Jiang and Conrath (1997), Lin (1998). All dictionary-based measures have the disadvantage of limited coverage: they cannot deal with many proper names and new words that are not in the dictionary. For WordNet-based approaches, there is the additional issue that they tend to work well only for nouns because the noun hierarchy in WordNet is the most developed. Also, most of the WordNet-based measures do not work for words with different part-of-speech, with small exceptions such as the extended Lesk measure (Banerjee and Pedersen, 2003).

We did a pre-screening of the various semantic similarity measures in order to choose the one measure of each type (dictionary-based and corpus-based) that seemed most promising for our task of detecting semantic outliers in automatic speech transcripts. The dictionary-based approaches that we evaluated were: the WordNet-based measure by Leacock and Chodorow (1987), and one other dictionary-based measure that uses the Roget thesaurus. The Roget measure (Jarmasz and Szpakowicz, 2003) has the advantage that it works across part-of-speech. The corpus-based measures we evaluated were: (a) the cosine measure based on word co-occurrence vectors (Lesk, 1969), (b) a new method that computes the Pearson correlation coefficient of the co-occurrence vectors instead of the cosine, and (c) a measure based on point-wise mutual information. We computed the first two measures on the 100-million-words British National Corpus (BNC)[1], and the third one on a much larger-corpus of Web data (one terabyte) accessed through the Waterloo Multitext system (Clarke and Terra, 2003). The reason for using corpora of different sizes is that PMI is the only one of the three corpus-based approaches that scales up to a terabyte corpus.

We describe here in detail the PMI corpus-based measure, because it is the most important for this paper. The semantic similarity score between two words w_1 and w_2 is defined as the probability of seeing the two words together divided by the probability of each word separately: $PMI(w_1,w_2) = \log [P(w_1,w_2) / (P(w_1)\cdot P(w_2))] = \log [C(w_1,w_2)\cdot N / (C(w_1)\cdot C(w_2))]$, where $C(w_1,w_2)$, $C(w_1)$, $C(w_2)$ are frequency counts, and N is the total number of words in the corpus. Such counts can easily and efficiently be retrieved for a terabyte corpus using the Waterloo Multitext system.

In order to assess how well the semantic similarity measures correlate with human perception, we use the set of 30 word pairs of Miller and Charles (1991), and the 65 pairs of Rubenstein and Goodenough (1965). Both used humans to judge the similarity. The Miller and Charles pairs were a subset of the Rubenstein and Goodenough pairs. Note that both of those sets were limited to nouns that appeared in the Roget thesaurus, and they are therefore favorably biased towards dictionary-based approaches. Table 1 shows the correlation of 5 similarity measures for the Rubenstein and Goodenough (R&G) and Miller and Charles (M&C) dataset. Note that although there are many WordNet-based semantic similarity measures, we only show correlations for Leacock and Chodorow (L&C) because it was previously shown to be better correlated (Jarmasz and Szpakowicz, 2003). We do not show figures for hybrid measures either because the same study showed L&C to be better.

Table 1: Correlation between human assigned and various machine assigned semantic similarity scores.

	Dictionary-based		**Corpus-based**		
	L&C	**Roget**	**Cos.**	**Corr.**	**PMI**
M&C	0.821	0.878	0.406	0.438	0.759
R&G	0.852	0.818	0.472	0.517	0.746

We see that the WordNet-based L&C measure based (Leacock and Chodorow, 1998 and the Roget measure (Jarmasz and Szpakowicz, 2003) both achieve high correlations but the two vector corpus-based measures (Cosine and Pearson Correlation) achieve much lower correlation. The only corpus-based measure that does well is PMI, probably because of the much larger corpus.

[1] http://www.natcorp.ox.ac.uk/index.html

We decided to experiment with two of the measures (one corpus-based and one thesaurus based) for computing the semantic similarity of word pairs in Step 2 of the algorithm described in Section 3. The two measures are: PMI computed on the Waterloo terabyte corpus and the Roget-based measure. These two seem the most promising given the nature of our task and the correlation figures reported above.

6 Evaluation Measures

We use several evaluation measures to determine how well our algorithm works for identifying semantic outliers. As summarized in Table 2, the task of detecting recognition errors can be viewed as a classification task. For each word, the algorithm must predict whether or not that word was transcribed correctly.

Table 2: Recognition error detection can be seen as a classification task.

	Correctly transcribed (actual)	NOT Correctly transcribed (actual)
Correctly transcribed (predicted)	True Positive (TP)	False Positive (FP)
NOT Correctly transcribed (predicted)	False Negative (FN)	True Negative (TN)

Note that we decide if a word is actually correctly transcribed or not by using the alignment of an automatic transcript with the manual transcript. A standard evaluation tool (sclite[2]) computes WER by counting the number of substitutions, deletions, and insertions needed to align a reference transcript with a hypothesis file. It also marks the words that are correct in automatic transcript (the hypothesis file). The rest of the words are the actual recognition errors (the insertions or substitutions). The deletions – words that are absent from the automatic transcript – cannot be tagged by the confidence measure.

We define the following performance measures in order to evaluate the improvement of the filtered transcripts compared to the initial transcripts:

1. **Word error rate** in the initial transcript and in the filtered transcript. These measures can be computed with and without stopwords (for which our algorithm does not apply). Note that WER without stopwords could be slightly lower than traditional WER mostly because content words tend to be recognized more accurately than stopwords (Désilets *et al.* 2001). When filtering out semantic outliers, there will be gaps in the filtered transcript, therefore the general WER might not improve because it penalizes heavily the deletions.

2. **Content word error rate (cWER)**. This is the error rate in an automatic transcript (initial or filtered) from the point of view of the confidence measure, for the content words only. It penalizes the words in the automatic transcripts that should not be there, but not any missing words (no deletions are penalized). In the case of a transcript filtered by our algorithm, it excludes not only the stopwords, but also the filtered words. We computed cWER with sclite without penalizing for the gaps created by the filtered words.

3. **The percentage of lost good content words (%Lost)**. This is the percentage of correctly recognized content words which are lost in the process of filtering out recognition errors, defined as: %Lost = 100 * FN / (TP + FN). We could also compute the **percent of discarded words**, without regard if they should have been filtered out or not. D = (TN + FN) / (TP + FP + TN + FN).

4. **Precision (P)**, **Recall (R)** and **F-measure**. **Precision** is the proportion of truly correct words contained in the list of content words which the algorithm labeled as correct. **Recall** is the proportion of truly correct content words that the algorithm was able to retain. **F-measure** is the geometric mean of P and R and expresses a tradeoff between those two measures. P = TP / (TP + FP); R = TP / (TP + FN); F = 2PR / (P+R).

7 Results

We ran various configurations of the algorithm described in Section 4 on the 100 story sample from the TDT2 corpus. This section discusses the results of those experiments. We studied the Precision-Recall (P-R) curves for various configurations of our algorithm over the 100 stories, for the two types of transcripts: the BBN and Dragon datasets. Figures 1 and 2 show an example for each dataset. Each point on a P-R curve shows the Precision and Recall for one value of K in {0, 20, 40, 60, 80,

[2] http://www.nist.gov/speech/tools/

100, 120, 140, 160, 180, 200}. Points on the left correspond to aggressive filtering (high values of K), whereas points on the right correspond to lenient filtering (low values of K).

First, we looked at the relative merits of the two semantic similarity measures (PMI and Roget) for Step 2. Figures 1 and 2 plot the P-R curves for the All-PMI-AVG and All-Roget-AVG configurations. The graphs clearly indicate that PMI performs better, especially for the high WER Dragon dataset. So PMI was used in the rest of the experiments.

Next, we looked at the variants for setting up the neighborhood N(w) in Step 1 (All vs. Window). The three P-R curves for All-PMI-X and Window-PMI-X for all aggregation approaches X in {AVG, MAX, 3MAX} are not shown here because they were similar to the P-PMI curves from Figures 1 and 2, for the BBN dataset and for the Dragon dataset, respectively. The Window variant was marginally better for X=MAX on both datasets, as well as for X=3MAX on the BBN dataset. In all other cases, the Window and All variants performed approximately the same.

Next, we looked at the different schemes for aggregating the pair-wise similarity scores in Step 3 (AVG, MAX, 3MAX). By plotting the P-R curves for All-PMI-AVG, All-PMI-MAX, and All-PMI-3MAX for both datasets we obtained again curves similar to the P-PMI curves from Figures 1 and 2. It seemed that AVG performs slightly better for high Recall, the difference being more marked when there is no windowing or when we are working on the Dragon dataset. The 3MAX and MAX variants seemed to be slightly better at high Precision with acceptable Recall values, with 3MAX being always equal or very slightly better than MAX. In an audio gisting and browsing context Precision is more important than Recall, therefore we can choose 3MAX.

Having established Window-PMI-3MAX as one of the better configurations, we now look more closely at its performance.

Figures 3 and 4 show how the content word error rate (cWER), the percentage of lost good words (%Lost), and the F-measure vary as we apply more and more aggressive error filtering (by increasing K) to both datasets. We see that our semantic outlier filtering approach is able to significantly reduce the number of transcription errors, while losing some correct words. For example, with the

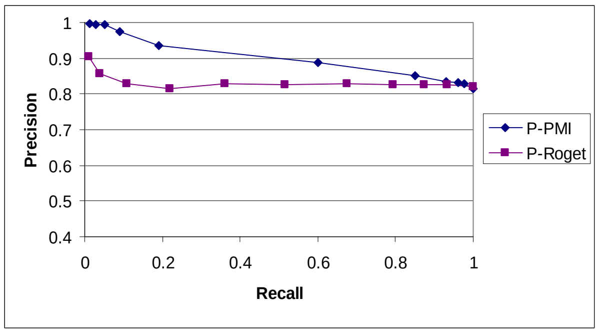


Fig 1: P-R curves of PMI vs. Roget (with All and AVG) on the BBN dataset. Each P-R point corresponds to a different value of the threshold K (high Recall for low values of K, high Precision for high values of K).

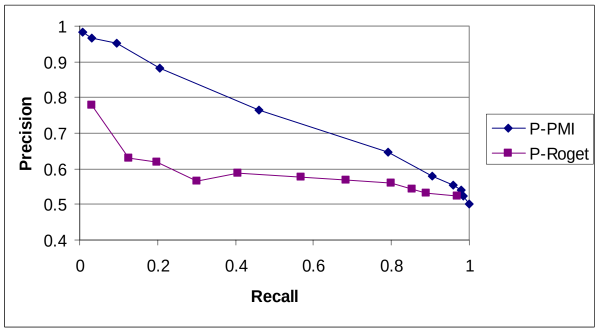


Fig 2: P-R curves of PMI vs. Roget (with All and AVG) on the Dragon dataset

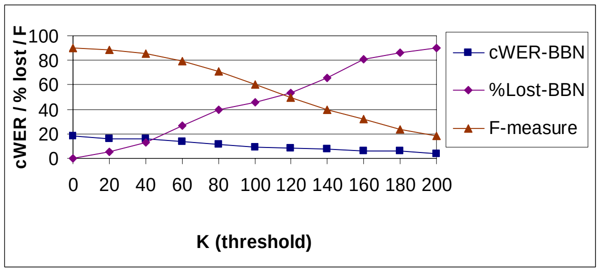


Fig.3. Content Words Error Rate (cWER), %Lost good keywords (%Lost) and F-measure as a function of the filtering level K for the Window-PMI-3MAXconfiguration on the BBN dataset.

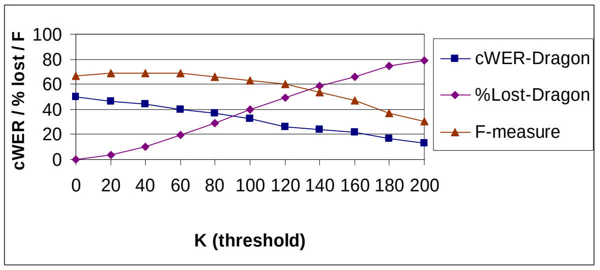


Fig.4. Content Words Error Rate (cWER), %Lost good keywords (%Lost) and F-measure as a function of the filtering level K for the Window-PMI-3MAX configuration on the Dragon dataset.

moderately accurate BBN dataset, we can reduce cWER by 50%, while losing 45% of the good content words (K=100). For the low accuracy Dragon dataset, we can reduce cWER by 50%, while losing 50% of the good content words (K=120). We can choose lower thresholds, for smaller reduction in cWER but smaller percent of lost good content words. Even small reductions in cWER are important, especially for less-than-broadcast conditions where WER is initially very high.

In general, we were not able to show an improvement in WER computed in a standard way (item 1 in Section 6), because of the high penalty due to deletions for both filtered semantic outliers and lost good content words. The percent of lost good words is admittedly too high, but this seems to be the case for speech error confidence measures (which do not remove the words tagged as incorrect). Also, for the purpose of audio browsing and gisting, we believe that fewer errors even with loss of content are preferable for intelligibility.

Comparing our results to those reported by Cox and Dasmahapatra (2000) our PMI-based measure seems to performs better than their LSA-based measure, judging by the shape of the Precision-Recall curves. (For example, at Precision=90%, they obtained Recall=12%, whereas we obtain 20%. At Precision=80%, they obtain Recall=50%, whereas we get Recall=100%.) Note however that their results and ours are not completely comparable since the experiments used different audio corpora (WSJCAM0 vs. TDT2), but those two corpora seem to exhibit similar initial WERs (the WER appears to be around 30% for WSJCAM0; the WER is 27.6% for our BBN dataset). Also, it is worth noting the LSA measure was computed based on a corpus that was very similar to the audio corpus used to evaluate the performance of the measure (both were Wall Street Journal corpora). If one was to evaluate this measure on audio from a completely different domain (ex: news in the scientific or technical domain), one would expect the performance to drop significantly. In contrast, our PMI measure was computed based on a general sample of the World Wide Web, which was not tailored to the audio corpus used to evaluate its performance. Therefore, our numbers are probably more representative of what would be experienced with audio corpora outside of the Wall Street Journal domain.

8 Conclusion and Future Work

We presented a basic method for filtering recognition errors of content words from automatic speech transcripts, by identifying semantic outliers. We described and evaluated several variants of the basic algorithm.

In future work, we plan to run our experiments on other datasets when they become available to us. In particular, we want to experiment with multi-topic audio documents where we expect more marked advantages for windowing and alternative aggregation schemes like MAX and 3MAX. We plan to explore ways to scale up other corpus-based semantic similarity measures to large terabyte corpora. We plan to explore more approaches to detecting semantic outliers, for example clustering or lexical chains (Hirst and St-Onge, 1997).

The most promising direction is to combine our method with confidence measures that use internal information from the ASR system (although the internal information is hard to obtain when using an ASR as a black box, and it could be recognizer-specific). A combination is likely to improve the performance, with the PMI-based measure contributing at the high-Precision end and the internal ASR measure contributing to the high-Recall end of the spectrum. To increase Recall we can also identify named entities and not filter them out. Some named entities could have high semantic similarity with the text if they are frequently mentioned in the same contexts in the Web corpus, but some names could be common to many contexts.

Another future direction will be to actually correct the errors instead of just filtering them out. For example, we might look at the top N speech recognizer hypotheses (for a fairly large N like 1000) and choose the one that maximizes semantic cohesion. A final direction for research is to conduct experiments with human subjects, to evaluate the degree to which filtered transcripts are better than unfiltered ones for tasks like browsing, gisting and searching audio clips.

Acknowledgments

We thank the following people: Peter Turney and his colleagues for useful feedback; Gerald Penn for feedback on earlier versions of this paper; Egidio Terra and Charlie Clarke for giving us permission to use the Multitext System, the NRC copy; Mario Jarmasz and Stan Szpakowicz for sharing their code for the Roget similarity measure; Aminul Islam for the correlation figures and the correlative measure. Our research is supported by the Natural Sciences and Engineering Research Council of Canada, University of Ottawa, IBM Toronto Centre for Advanced Studies, and the National Research Council.

References

Alexander Budanitsky and Graeme Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. *Workshop on WordNet and Other Lexical Resources, NAACL 2001,* Pittsburgh, PA, USA, 29-34.

Satanjeev Banerjee, and Ted Pedersen. 2003. Gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, Acapulco, Mexico.

Charlie Clarke and Egidio Terra. 2003. Passage retrieval vs. document retrieval for factoid question answering. *ACM SIGIR'03*, 327-328.

Stephen Cox and Srinandan Dasmahapatra. 2000. A Semantically-Based Confidence Measure for Speech Recognition, *Int. Conf. on Spoken Language Processing*, Beijing, China, vol. 4, 206-209.

Stephen Cox and R.C. Rose. 1996. Confidence Measures for the SWITCHBOARD Database. *IEEE Conf. on Acoustics, Speech, and Signal Processing,* 511-515.

Lin Chase. 1997. Word and Acoustic Confidence Annotation for Large Vocabulary Speech Recognition, *Proceedings of Eurospeech'97*, Rhodes, Greece, 815-818.

Alain Désilets, Berry de Brujin, and Joel Martin. 2001. Extracting keyphrases from spoken audio documents. *SIGIR'01 Workshop on Information Retrieval Techniques for Speech Applications*, 36-50.

Diana Inkpen and Alain Désilets. 2004. Extracting semantically-coherent keyphrases from speech. *Canadian Acoustics,* 32(3):130-131.

L.Gillick, Y.Ito, and J.Young. 1997. A Probabilistic Approach to Confidence Estimation and Evaluation. *IEEE Conf. on Acoustics, Speech, and Signal Processing,* 266-277.

Julia Hirschberg, Steve Whittaker, Donald Hindle, Fernando Pereira, Amit Singhal. 1999. Finding information in audio: a new paradigm for audio browsing and retrieval. *Proceedings of the ESCA ETRW Workshop,* 26-33.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In: C. Fellbaum (editor), *WordNet: An electronic lexical database and some of its applications*, The MIT Press, Cambridge, MA, 305-332.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's thesaurus and semantic similarity, *Proceedings of the International Conference RANLP-2003 (Recent Advances in Natural Language Processing)*, Borovets, Bulgaria, 212-219.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X),* Taiwan.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: representation of knowledge. *Psychological Review* 104: 211-240.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. In C. Felbaum (editor), *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, MA, 264-283.

M.E. Lesk. 1969. Word-word associations in document retrieval systems. *American Documentation* 20(1): 27-38.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference of Machine Learning.*

Changxue Ma, Mark A. Randolph, and Joe Drish. 2001. A support vector machines-based rejection technique for speech recognition. *Proceedings of ICASSP'01*, Salt Lake City, USA, vol. 1, 381-384.

Lidia Mangu and M. Padmanabhan. 2001. Error corrective mechanisms for speech recognition. *Proceedings of ICASSP'01*, Salt Lake City, USA, vol. 1, 29-32.

George A. Miller and W.G. Charles. 1991. Contextual correlates of semantic similarity, *Language and Cognitive Processes*, 6(1):1-28.

Christine Nakatani, Steve Whittaker, Julia Hirshberg. 1998. Now you hear it, now you don't: Empirical Studies of Audio Browsing Behavior. *Proceedings of the Fifth International Conference on Spoken Language Processing, (SLP'98),* Sydney, Australia.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th Joint International Conference of Artificial Intelligence*, Montreal, Canada, 448-453.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of ACM*, 8(10): 627-633.

Thomas Schaaf and Thomas Kemp. 1997. Confidence measures for spontaneous speech recognition, in *Proceedings of ICASSP'97*, Munich, Germany, vol. II, 875-878.

Gabriel Skantze and J. Edlund. 2004. *Error detection on word level*. In Proceedings of Robust 2004, Norwich.

Peter D. Turney. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL, *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, Freiburg, Germany, 491-502.

Lina Zhou, Jinjuan Feng, Andrew Sears, Yongmei Shi. 2005. Applying the Naïve Bayes Classifier to Assist Users in Detecting Speech Recognition Errors. *Procs. of the 38th Annual Hawaii International Conference on System Sciences).*

Z.Y. Zhou and Helen M. Meng, 2004. A Two-Level Schema for Detecting Recognition Errors, *Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP)*, Korea.

Redundancy-based Correction of Automatically Extracted Facts

Roman Yangarber and **Lauri Jokipii**
Department of Computer Science
University of Helsinki, Finland
`first.last@cs.helsinki.fi`

Abstract

The accuracy of event extraction is limited by a number of complicating factors, with errors compounded at all sages inside the Information Extraction pipeline. In this paper, we present methods for recovering automatically from errors committed in the pipeline processing. Recovery is achieved via post-processing facts aggregated over a large collection of documents, and suggesting corrections based on evidence external to the document. A further improvement is derived from propagating multiple, *locally non-best* slot fills through the pipeline. Evaluation shows that the global analysis is over 10 times more likely to suggest valid corrections to the local-only analysis than it is to suggest erroneous ones. This yields a substantial overall gain, with no supervised training.

1 Introduction

Information Extraction (IE) is a technology for finding facts in plain text, and coding them in a logical representation, such as, e.g., a relational database. IE is typically viewed and implemented as a sequence of stages—a "pipeline":

1. Layout, tokenization, lexical analysis
2. Name recognition and classification
3. Shallow (commonly,) syntactic parsing
4. Resolution of co-reference among entities
5. Pattern-based event matching and role mapping
6. Normalization and output generation

While accuracy at the lowest levels can reach high 90's, as the stages advance, complexity increases and performance degrades considerably.

The problem of IE as a whole, as well each of the listed subproblems, has been studied intensively for well over a decade, in many flavors and varieties. Key observations about much state-of-the-art IE are:

a. IE is typically performed by a pipeline process;

b. Only one hypothesis is propagated through the pipeline for each fact—the "best guess" the system can make for each slot fill;

c. IE is performed in a document-by-document fashion, applying *a priori* knowledge locally to each document.

The *a priori* knowledge may be encoded in a set of rules, an automatically trained model, or a hybrid thereof. Information extracted from documents—which may be termed *a posteriori* knowledge—is usually not reused across document boundaries, because the extracted facts are imprecise, and are therefore not a reliable basis for future extraction.

Furthermore, locally non-best slot fills are not propagated through the pipeline, and are consequently not available downstream, nor for any global analysis.

In most systems, these stages are performed in sequence. The locally-best slot fills are passed from the "lower-" to the "higher-level" modules, without feedback. Improvements are usually sought (e.g., the ACE research programme, (ACE, 2004)) by boosting performance at the lower levels, to reap benefits in the subsequent stages, where fewer errors are propagated.

The point of departure for this paper is: the IE process is noisy and imprecise at the single-document level; this has been the case for some time, and though there is much active research in the area, the situation is not likely to change radically in the immediate future—rather, we can expect slow, incremental improvements over some years.

In our experiments, we approach the performance problem from the opposite end: start with the extracted results and see if the totality of *a posteriori* knowledge about the domain—knowledge generated by the same noisy process we are trying to improve—can help recover from errors that stem from locally insufficient *a priori* knowledge.

The aim of the research presented in this paper is to improve performance by aggregating related facts, which were extracted from a large document collection, and to examine to what extent the correctly extracted facts can help correct those that were extracted erroneously.

The rest of the paper is organized as follows. Section 2 contains a brief review of relevant prior work. Section 3 presents the experimental setup: the text corpus, the IE process, the extracted facts, and what aspects of the the extracted facts we try to improve in this paper. Section 4 presents the methods for improving the quality of the data using global analysis, starting with a naive, baseline method, and proceeding with several extensions. Each method is then evaluated, and the results are examined in section 5. In section 6, we present further extensions currently under research, followed by the conclusion.

2 Prior Work

As we stated in the introduction, typical IE systems consist of modules arranged in a cascade, or a pipeline. The modules themselves are be based on heuristic rules or automatically trained, there is an abundance of approaches in both camps (and everywhere in between,) to each of the pipeline stages listed in the introduction.

It is our view that to improve performance we ought to depart from the traditional linear, pipeline-style design. This view is shared by others in the research community; the potential benefits have previously been recognized in several contexts.

In (Nahm and Mooney, 2000a; Nahm and Mooney, 2000b), it was shown that learning rules from a fact base, extracted from a corpus of job postings for computer programmers, improves future extraction, even though the originally extracted facts themselves are far from error-free. The idea is to mine the data base for association rules, and then to integrate these rules into the extraction process.

The baseline system is obtained by supervised learning from a few hundred manually annotated examples. Then the IE system is applied to successively larger sets of unlabeled examples, and association rules are mined from the extracted facts. The resulting combined system (trained model plus association rules) showed an improvement in performance on a test set, which correlated with the size of the unlabeled corpus.

In work on improving (Chinese) named entity tagging, (Ji and Grishman, 2004; Ji and Grishman, 2005), show benefits to this component from integrating decisions made in later stages, viz. coreference, and relation extraction.[1]

Tighter coupling and integration between IE and KDD components for mutual benefit is advocated by (McCallum and Jensen, 2003), which present models based on CRFs and supervised training.

This work is related in spirit to the work presented in this paper, in its focus on leveraging cross-document information that information—though it is inherently noisy—to improve local decisions. We expect that the approach could be quite powerful when these ideas are used in combination, and our experiments seem to confirm this expectation.

3 Experimental Setup

In this section we describe the text corpus, the underlying IE process, the form of the extracted facts, and the specific problem under study—i.e., which aspects of these facts we first try to improve.

[1]Performance on English named entity tasks reaches mid to high 90's in many domains.

3.1 Corpus

We conducted experiments with redundancy-based auto-correction over a large database of facts extracted from the texts in ProMED-Mail, a mailing list which carries reports about outbreaks of infectious epidemics around the world and the efforts to contain them. This domain has been explored earlier; see, e.g., (Grishman et al., 2003) for an overview.

Our underlying IE system is described in (Yangarber et al., 2005). The system is a hybrid automatically- and manually-built pattern base for finding facts, an HMM-based name tagger, automatically compiled and manually verified domain-specific ontology, based in part on MeSH, (MeS, 2004), and a rule-based co-reference module, that uses the ontology.

The database is live on-line, and is continuously updated with new incoming reports; it can be accessed at `doremi.cs.helsinki.fi/plus/`.

Text reports have been collected by ProMED-Mail for over 10 years. The quality of reporting (and editing) has been rising over time, which is easy to observe in the text data. The distribution of the data, aggregated by month is shown in Figure 1, where one can see a steady increase in volume over time.[2]

3.2 Extracted Facts

We now describe the makeup of the data extracted from text by the IE process, with basic terminology.

Each document in the corpus, contains a single *report*, which may contain one or more *stories*. Story breaks are indicated by layout features, and are extracted by heuristic rules, tuned for this domain and corpus. When processing a multi-story report, the IE system treats each story as a separate document; no information is shared among stories, except that the text of the main headline of a multi-story report is available to each story. [3]

Since outbreaks may be described in complex ways, it is not obvious how to represent a single fact in this context. To break down this problem, we use the notion of an *incident*. Each story may contain

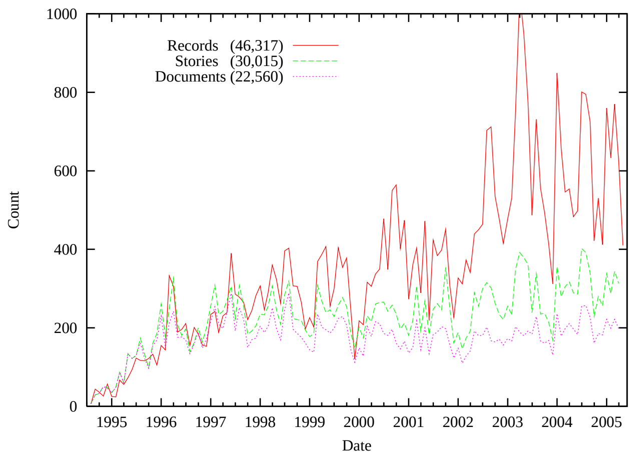


Figure 1: Distribution of data in ProMED-Mail

multiple outbreak-related incidents/facts.[4]

We analyze an outbreak as a series of incidents. The incidents may give "redundant" information about an outbreak, e.g., by covering overlapping time intervals or geographic areas. For example, a report may first state the number of cases within the last month, and then give the total for the entire year. We treat each of these statements as a separate incident; the containment relations among them are beyond the scope of our current goals.[5]

Thus each incident corresponds to a partial description of an outbreak, over a period of time and geographic area. This makes it easy to represent each incident/fact as a separate row in the table.

The key fields of the incident table are:

- Disease Name
- Location
- Date (start and end)

Where possible, the system also extracts information about the victims affected in the incident—their count, severity (affected or dead), and a descriptor (people, animals, etc.). The system also extracts bookkeeping information about each incident: locations of mentions of the key fields in the text, etc.

The system's performance is currently at 71.16 F-measure: 67% recall, 74% precision. This score is obtained by a MUC scorer (Douthat, 1998) on a 50-document test corpus, which was manually tagged with correct incidents with these slots. We have

[2]This is beneficial to the IE process, which operates better with formulaic, well-edited text.

[3]The format of the documents in the archive can be examined by browsing the source site `www.promedmail.org`.

[4]In this paper, we use the terms *fact*, *incident*, and *event* interchangeably.

[5]This problem is addressed in, e.g., (Huttunen et al., 2002).

no blind-test corpus at present, but prior experience suggests that we ought to expect about a 10% reduction in F-measure on unseen data; this is approximately borne out by our informal evaluations.

Further, the system attempts to "normalize" the key fields. An alias for a disease name (e.g., "bird flu") is mapped to a *canonical* name ("avian influenza.")[6] Date expressions are normalized to a standard format `yyyy.mm.dd`–`yyyy.mm.dd`.[7]

Note that the system may not be able to normalize some entities, which then remain un-normalized.

Such normalization is clearly helpful for searching, but it is not only a user-interface issue. Normalizing reduces sparseness of data; and since our intent is to aggregate related facts across a large fact base, excessive variation in the database fields would reduce the effectiveness of the proposed methods.

3.3 Experimental Focus: Location Normalization

A more complex problem arises out of the need to normalize location names. For each record, we normalize the location field—which may be a name of a small village or a larger area—by relating it to the name of the containing country; we also decided to map locations in the United States to the name of the containing state, (rather than the name of the country, "USA").[8] This mapping will be henceforth referred to as "location–state," for short. The ideas presented in the introduction are explored in the remainder of this paper in the context of correcting the location–state mapping.

Section 6 will touch upon our current work on extending the methodology to slots other than *state*. (Please see Section 5 for further justification of this choice for our initial experiments.)

To make the experiments interesting and fair, we kept the size of the gazetteer small. The *a priori* geographic knowledge base contains names of countries of the world (270), with aliases for several of them; a list of capitals and other selected major cities (300); a list of states in the USA and acronyms (50); major US cities (100); names of the (sub)continents (10), and oceans. In our current implementation, continents are treated semantically as "states" as well.[9]

The IE system operates in a local, document-by-document fashion. Upon encountering a location name that is not in its dictionaries, the system has two ways to map it to the state name. One way is by matching patterns over the immediate local context, ("Milan, Italy"). Failing that, it tries to find the corresponding state by positing an "underspecified" state name (as if referred to by a kind of special "pronoun") and mapping the location name to that. The reference resolution module then finds the most likely antecedent entity, of the semantic type "state/country," where likelihood is determined by its proximity to the mention of the location name.

Note that the IE system outputs only a single, best hypothesis for the *state* fill for each record.

3.4 The Data

The database currently contains about $46,317$ individual facts/incidents, extracted from $30,015$ stories, from $22,560$ reports (cf. Fig. 1). Each incident has a *location* and a *state* filler. We say a location name is "ambiguous" if it appears in the *location* slot of at least two records, which have different names in the *state* slot. The number of distinct "ambiguous" location names is $1,020$.

Note, this terminology is a bit sloppy: the fillers to which we refer as "ambiguous location names", may not be valid location names at all; they may simply be errors in the IE process. E.g., at the name classification stage, a disease name (especially if not in the disease dictionary) may be misclassified, and used as a filler for the *location* slot.

We further group together the *location* fills by stripping lower-case words that are not part of the proper name, from the front and the end of the fill. E.g., we group together "southern Mumbai" and "the Mumbai area," as referring to the same name.

After grouping and trimming insignificant words, the number of distinct names appearing in *location* fills is 600, which covers a total of 6583 records, or 14.2% of all extracted facts. As an estimate of the *potential* for erroneous mapping from locations to states, this is quite high, about 1 in 7 records.[10]

[6]This is done by means of a set of scenario-specific patterns and a dictionary of about 2500 disease names with aliases.

[7]Some date intervals may not have a starting date, e.g., if the text states "As of last Tuesday, the victim count is N..."

[8]This decision was made because otherwise records with state = USA strongly skew the data, and complicate learning.

[9]By the same token, both *Connecticut* and *USA* are "states."

[10]Of course, it can be higher as well, if the IE system con-

4 Experiments and Results

We now present the methods of correcting possible errors in the location–state relation. A method M tries to suggest a new value for the *state* fill for every incident $\mathbf{I}$ that contains an ambiguous *location* fill:

$$NewState(\mathbf{I}) = \arg \max_{s \in \mathcal{S}_M(\mathbf{I})} Score_M(s, \mathbf{I}) \quad (1)$$

where $\mathcal{S}_M(\mathbf{I})$ is a set of all candidate states considered by M for $\mathbf{I}$; $Score_M(s, \mathbf{I})$ is the scoring function specific to M. The method chooses the candidate state which maximizes the score.

For each method below, we discuss how $\mathcal{S}_M$ and $Score_M$ are constructed.

4.1 Baseline: Raw Majority

We begin with a simple recovery approach. We simply assume that the correct state for an ambiguous location name is the state most frequently associated with it in the database. We denote by $\mathcal{D}$ the set of all incidents in the database. For an incident $\mathbf{I} \in \mathcal{D}$, we write $l, s \sqsubset \mathbf{I}$ when location l, state s, etc., "belong" to $\mathbf{I}$, i.e., are extracted as fills in $\mathbf{I}$. In the baseline method, B, for each incident $\mathbf{I}$ where $l \sqsubset \mathbf{I}$ is one of the 600 ambiguous location names, we define:

$$\mathcal{S}_B(\mathbf{I}) = \{s' : \exists \mathbf{I}' \in \mathcal{D} \mid (l, s') \sqsubset \mathbf{I}'\}$$
$$Score_B(s', \mathbf{I}) = |\{\mathbf{I}' \in \mathcal{D} \mid (l, s') \sqsubset \mathbf{I}'\}|$$

i.e., s' is a candidate if it is a *state* fill in some incident whose *location* fill is also l; the score is the number of times the pair (l, s') appear together in some incident in $\mathcal{D}$. The majority winner is then suggested as the "correct" state, for *every* record containing l. By "majority" winner we mean the candidate with the *maximal* count, rather than a state with more than half of the votes. When the candidates tie for first place, no suggestions are made—although it is quite likely that some of the records carrying l will have incorrect *state* fills.

A manual evaluation of the performance of this method is shown in Table 1, the *Baseline* column.

The first row shows for how many records this method suggested a change from the original, IE-filled *state*. The baseline changed 858 incidents. This constitutes about 13% out of the maximum number of changeable records, 6583.

Thus, this number represents the volume of the potential gain or loss from the global analysis: the proportion of records that actually get modified.

The remaining records were unchanged, either because the majority winner coincides with the original IE-extracted state, or because there was a tie for the top score, so no decision could be made.

We manually verified a substantial sample of the modified records. When verifying the changes, we referred back to the text of the incident, and, when necessary, consulted further geographical sources to determine exactly whether the change was correct in each case.

For the baseline we had manually verified 27.7% of the changes. Of these, 68.5% were a clear gain: an incorrect state was changed to a correct state. 6.3% were a clear loss, a correct state lost to an incorrect one. This produces quite a high baseline, surprisingly difficult to beat.

The next two rows represent the "grey" areas. These are records which were difficult to judge, for one of two technical reasons. A: the "location" name was itself *erroneous*, in which case these redundancy-based approaches are not meaningful; or, B: the suggestion replaces an area by its sub-region or super-region, e.g., changing "Connecticut" to "USA", or "Europe" to "France."[11]

Although it is not strictly meaningful to judge whether these changes constitute a gain or a loss, we nonetheless tried to assess whether changing the state hurt the accuracy of *the incident*, since the incident may have a correct state even though its location is erroneous (case A); likewise, it may be correct to say that a given location is indeed a part of Connecticut, in which case changing it to USA loses information, and is a kind of loss.

That is the interpretation of the grey gain and loss instances. The final row, *no loss*, indicates the proportion of cases where an originally incorrect state name was changed to a new one, also incorrect.

sistently *always* maps some location name to the same wrong state; these cases are below the radar of our scheme, in which the starting point is the "ambiguous" locations.

[11]Note, that for some locations, which are not within any one state's boundary, a continent is a "correct state", for example, "the Amazon Region," or "Serengeti."

Records	*Baseline*		*DB-filtered*		*Confidence*		*Multi-candidate*	
Changed	13.0%	858/6583	8.7%	577/6583	9.7%	642/6583	**16.2**%	1072/6583
Verified	27.7%	238/858	38.6%	223/577	37.1%	238/642	26.5%	284/1072
Gain	68.5%	163/238	71.3%	159/223	80.3%	191/238	**76.4**%	217/284
Loss	6.3%	15/238	2.2%	5/223	1.3%	3/238	**3.5**%	10/284
Grey gain	10.9%	26/238	12.6%	28/223	11.8%	28/238	13.7%	39/284
Grey loss	6.7%	16/238	5.4%	12/223	0.0%	0/238	2.1%	6/284
No loss	7.6%	18/238	8.5%	19/223	6.7%	16/238	4.2%	12/284

Table 1: Performance of Correction Methods

4.2 Database Filtering

Next we examined a variant of baseline raw majority vote, noting that simply choosing the state most frequently associated with a location name is a bit naive: the location–state relation is not functional—i.e., some location names map to more than one state in reality. There are many locations which share the same name.[12]

To approach this more intelligently, we define:

$$\mathcal{S}_F(\mathbf{I}) = \mathcal{S}_B(\mathbf{I}) \cap StatesInStory(\mathbf{I})$$
$$Score_F(s', \mathbf{I}) = Score_B(s', \mathbf{I})$$

The baseline vote counting across the data base (DB) produced a ranked list of candidate states s' for the location $l \sqsubset \mathbf{I}$. We then filtered this list through $StatesInStory(\mathbf{I})$, the list of states mentioned in the story containing the incident $\mathbf{I}$. The filtered majority winner was selected as the suggested change.

For example, the name “Athens” may refer to the city in Greece, or to the city in Georgia (USA). Suppose that Greece is the raw majority winner. The baseline method will always tag all instances of Athens as being in Greece. However, in a story about Georgia, Greece will likely not be mentioned at all, so it is safe to rule it out. This helps a minority winner, when the majority is not present in the story.

Surprisingly, this method did not yield a substantial improvement over the baseline, (though it was more careful by changing fewer records). This may indicate that NWP is not an important source of errors here: though many truly ambiguous locations do exist, they do not account for many instances in this DB.

[12]We refer to this as the “New-World phenomenon” (NWP), due to its prevalence in the Americas: “Santa Cruz” occurs in several Latin American countries; locations named after saints are common. In the USA, city and county names often appear in multiple states—Winnebago County, Springfield; many cities are named after older European cities.

4.3 Confidence-Based Ranking

A more clear improvement over the baseline is obtained by taking the *local confidence* of the state–location association into account. For each record, we extend the IE analysis to produce a confidence value for the state. Confidence is computed by simple, document-local heuristics, as follows:

If the location and state are both within the span of text covered by the incident—text which was actually matched by a rule in the IE system,—or if the state is the *unique* state mentioned in the story, it gets a score of 2—the incident has high confidence in the state. Otherwise, if the state is outside the incident's span, but is inside the same sentence as the incident, and is also the unique state mentioned in that sentence, it gets a score of 1. Otherwise it receives a score of zero.

Given the confidence score for each (location l, state s) pair, the majority counting is based on the *cumulative confidence*, $conf_{state}$ (l, s) in the DB, rather than on the *cumulative count* of occurrences of this pair in the DB:

$$\mathcal{S}_C(\mathbf{I}) = \mathcal{S}_F(\mathbf{I})$$
$$Score_C(s', \mathbf{I}) = \sum_{\mathbf{I}' \in \mathcal{D} | (l, s') \sqsubset \mathbf{I}'} conf_{state}(\mathbf{I}')$$

Filtering through the story is also applied, as in the previous method. The resulting method favors more correct decisions, and fewer erroneous ones.

We should note here, that the notion of confidence of a fill (here, the state fill) is naturally extended to the notion of confidence of a *record*: For each of the three key fills—location, date, disease name—compute a confidence based on the same heuristics. Then we say that a record **I** has high confidence, if it has non-zero confidence in all three of the key fills. The notion of record confidence is used in Section 6.

4.4 Multi-Candidate Propagation

Finally, we tried propagating multiple candidate state hypotheses for each instance of an ambiguous location name l:

$$\mathcal{S}_+(\mathbf{I}) = \bigcup_{\mathbf{I}' \in \mathcal{D} | l \sqsubset \mathbf{I}'} StatesInStory(\mathbf{I}')$$

$$Score_+(s', \mathbf{I}) = \sum_{\mathbf{I}' \in \mathcal{D} | l \sqsubset \mathbf{I}'} prox(s', \mathbf{I}')$$

where the proximity is inversely proportional to the distance of s' from incident $\mathbf{I}'$, in the story of $\mathbf{I}'$:

$$prox(s, \mathbf{I}) = \begin{cases} \frac{1}{Z(\mathbf{I})} \cdot \frac{1}{\Delta(s, \mathbf{I}) + 1} & if\ s \sqsubset \mathbf{I} \\ 0 & otherwise \end{cases}$$

For an incident **I** mentioning location l, the IE system outputs the list of all states $\{s\}$ mentioned in the same story; we then rank each s according to the inverse of *distance* Δ: the number of sentences between **I** and s. $Z(\mathbf{I})$ is a normalization factor.

The proximity for each pair (l, s), is between 0 and 1. Rather than giving a full point to a single, locally-best guess among the s's, this point is shared proportionately among all competing s's. For example, if states s_0, s_1, s_5 are in the same sentence as **I**, one, and five sentences away, respectively, then $Z(\mathbf{I}) = 1 + \frac{1}{2} + \frac{1}{6} = \frac{5}{3}$, and $prox(s_0) = 1 \cdot \frac{3}{5} = \frac{3}{5}$, $prox(s_1) = \frac{1}{2} \cdot \frac{3}{5} = \frac{3}{10}$,and $prox(s_5) = \frac{1}{6} \cdot \frac{3}{5} = \frac{1}{10}$.

The score for each state s for the given l is then the sum of proximities of s to l across all stories.

The resulting performance is substantially better than the baseline, while the *number of changed* records is substantially higher than in the competing methods. This is due to the fact that this method allows for a much larger pool of candidates than the others, and assigns to them much smoother weights, virtually eliminating ties in the ranking among hypotheses.

5 Discussion

Among the four competing approaches presented above, the baseline performs surprisingly well. We should note that this research is not aimed specifically at improving geographic Named Entity resolution. It is the first in a series of experiments aiming to leverage redundancy across a large fact base extracted from text, to improve the quality of extracted data. We chose to experiment with this relation first because of its simplicity, and because the *state* field is a key field in our application.

For this reason, the *a priori* geographic knowledge base was intentionally not as extensive as it might have been, had we tried in earnest to match locations with corresponding states (e.g., by incorporating the CIA Factbook, or other gazetteer).

The intent here is to investigate how a relation can be improved by leveraging redundancy across a large body of records. The support we used for geographic name resolution was therefore deliberately modest, cf. Section 3.3.

It is quite feasible to enumerate the countries and the larger regions, since they number in the low hundreds, whereas there are many tens of thousands of cities, towns, villages, regions, districts, etc.

6 Current Work

Three parallel lines of current research are:

1. combining evidence from multiple features
2. applying redundancy-based correction to other fields in the database
3. back-propagation of corrected results, to repair components that induced incorrect information.

The results so far presented show that even a naive, intuitive approach can help correct local errors via global analysis. We are currently working on more complex extensions of these methods.

Each method exploits one main feature of the underlying data: the distance from candidate state to the mention of the location name. In the multi-candidate hypothesis method, this distance is exploited explicitly in the scoring function. In the other methods, it is used inside the co-reference module of the IE pipeline, to find the (single) locally-best state.

However, other textual features of the state candidate should contribute to establishing the relations to a location mention, besides the raw distance. For example, at a given distance, it is very important whether the state is mentioned before the location (more likely to be related) vs. after the location (less likely). Another important feature: is the state mentioned in the main story/report headline? If so, its score should be raised. It is quite common for documents to declaim the focal state only once in the headline, and never mention it again, instead mentioning other states, neighboring, or otherwise relevant to the story. The distance measure used alone may be insufficient in such cases.

How are these features to be combined? One path is to use some combination of features, such as a weighted sum, with parameters trained on a manually tagged data set. As we already have a reasonably sized set tagged for evaluation, we can split it into two, train the parameter on a larger portion, evaluate on a smaller one, and cross-validate.

We will be using this approach as a baseline. However, we aim to use a much larger set of data to train the parameters, without manually tagging large training sets.

The idea is to treat the set of incidents with high *record confidence*, Sec. 4.3, rather than manually tagged data, as ground truth. Again, there "confident" truth will not be completely error-free, but because error rates are lower among the confident records, we may be able to leverage global analysis to produce the desired effect: training parameters for more complex models—involving multiple features—for global re-ranking of decisions.

Conclusion

Our approach rests on the idea that evidence aggregated across documents should help resolve difficult problems at the level of a given document.

Our experiments confirm that aggregating global information about related facts, and propagating locally non-best analyses through the pipeline, provide powerful sources of additional evidence, which are able to reverse incorrect decisions, based only on local and *a priori* information.

The proposed approach requires no supervision or training of any kind. It does, however require a substantial collection of evidence across a large body of extracted records; this approach needs a "critical mass" of data to be effective. Although large volume of facts is usually not reported in classic IE experiments, obtaining high volume should be natural in principle.

Acknowledgement

We'd like to thank Winston Lin of New York University, who worked on an early version of these experiments.

References

2004. Automatic content extraction.

A. Douthat. 1998. The Message Understanding Conference scoring software user's manual. In *Proc. 7th Message Understanding Conf. (MUC-7)*, Fairfax, VA.

R. Grishman, S. Huttunen, and R. Yangarber. 2003. Information extraction for enhanced access to disease outbreak reports. *J. of Biomed. Informatics*, **35**(4).

S. Huttunen, R. Yangarber, and R. Grishman. 2002. Complexity of event structure in information extraction. In *Proc. 19th Intl. Conf. Computational Linguistics (COLING 2002)*, Taipei.

H. Ji and R. Grishman. 2004. Applying coreference to improve name recognition. In *Proc. Reference Resolution Wkshop, (ACL-2004)*, Barcelona, Spain.

H. Ji and R. Grishman. 2005. Improving name tagging by reference resolution and relation detection. In *Proc. ACL-2005*, Amherst, Mass.

A. McCallum and D. Jensen. 2003. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *IJCAI'03 Workshop on Learning Statistical Models from Relational Data*.

2004. Medical subject headings.

U. Y. Nahm and R. Mooney. 2000a. A mutually beneficial integration of data mining and information extraction. In *AAAI-2000*, Austin, TX.

U. Y. Nahm and R. Mooney. 2000b. Using information extraction to aid the discovery of prediction rules from text. In *KDD-2000 Text Mining Wkshop*, Boston, MA.

R. Yangarber, L. Jokipii, A. Rauramo, and S. Huttunen. 2005. Extracting information about outbreaks of infectious epidemics. In *Proc. HLT-EMNLP 2005 Demonstrations*, Vancouver, Canada.

NeurAlign: Combining Word Alignments Using Neural Networks

Necip Fazil Ayan, Bonnie J. Dorr and **Christof Monz**
Department of Computer Science
University of Maryland
College Park, MD 20742
{nfa,bonnie,christof}@umiacs.umd.edu

Abstract

This paper presents a novel approach to combining different word alignments. We view word alignment as a pattern classification problem, where alignment combination is treated as a classifier ensemble, and alignment links are adorned with linguistic features. A neural network model is used to learn word alignments from the individual alignment systems. We show that our alignment combination approach yields a significant 20-34% relative error reduction over the best-known alignment combination technique on English-Spanish and English-Chinese data.

1 Introduction

Parallel texts are a valuable resource in natural language processing and essential for projecting knowledge from one language onto another. Word-level alignment is a critical component of a wide range of NLP applications, such as construction of bilingual lexicons (Melamed, 2000), word sense disambiguation (Diab and Resnik, 2002), projection of language resources (Yarowsky et al., 2001), and statistical machine translation. Although word-level aligners tend to perform well when there is *sufficient* training data, the quality decreases as the size of training data decreases. Even with large amounts of training data, statistical aligners have been shown to be susceptible to mis-aligning phrasal constructions (Dorr et al., 2002) due to many-to-many correspondences, morphological language distinctions, paraphrased and free translations, and a high percentage of function words (about 50% of the tokens in most texts).

This paper presents a novel approach to alignment combination, *NeurAlign*, that treats each alignment system as a black box and merges their outputs. We view word alignment as a pattern classification problem and treat alignment combination as a *classifier ensemble* (Hansen and Salamon, 1990; Wolpert, 1992). The ensemble-based approach was developed to select the best features of different learning algorithms, including those that may not produce a globally optimal solution (Minsky, 1991).

We use neural networks to implement the classifier-ensemble approach, as these have previously been shown to be effective for combining classifiers (Hansen and Salamon, 1990). Neural nets with 2 or more layers and non-linear activation functions are capable of learning any function of the feature space with arbitrarily small error. Neural nets have been shown to be effective with (1) high-dimensional input vectors, (2) relatively sparse data, and (3) noisy data with high within-class variability, all of which apply to the word alignment problem.

The rest of the paper is organized as follows: In Section 2, we describe previous work on improving word alignments and use of classifier ensembles in NLP. Section 3 gives a brief overview of neural networks. In Section 4, we present a new approach, *NeurAlign*, that learns how to combine individual word alignment systems. Section 5 describes our experimental design and the results on English-Spanish and English-Chinese. We demonstrate that NeurAlign yields significant improvements over the best-known alignment combination technique.

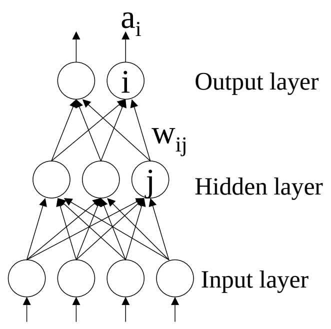


Figure 1: Multilayer Perceptron Overview

2 Related Work

Previous algorithms for improving word alignments have attempted to incorporate additional knowledge into their modeling. For example, Liu (2005) uses a log-linear combination of linguistic features. Additional linguistic knowledge can be in the form of part-of-speech tags. (Toutanova et al., 2002) or dependency relations (Cherry and Lin, 2003). Other approaches to improving alignment have combined alignment models, e.g., using a log-linear combination (Och and Ney, 2003) or mutually independent association clues (Tiedemann, 2003).

A simpler approach was developed by Ayan et al. (2004), where word alignment outputs are combined using a linear combination of feature weights assigned to the individual aligners. Our method is more general in that it uses a neural network model that is capable of learning nonlinear functions.

Classifier ensembles are used in several NLP applications. Some NLP applications for classifier ensembles are POS tagging (Brill and Wu, 1998; Abney et al., 1999), PP attachment (Abney et al., 1999), word sense disambiguation (Florian and Yarowsky, 2002), and parsing (Henderson and Brill, 2000).

The work reported in this paper is the first application of classifier ensembles to the word-alignment problem. We use a different methodology to combine classifiers that is based on *stacked generalization* (Wolpert, 1992), i.e., learning an additional model on the outputs of individual classifiers.

3 Neural Networks

A multi-layer perceptron (MLP) is a feed-forward neural network that consists of several units (neurons) that are connected to each other by weighted links. As illustrated in Figure 1, an MLP consists of one input layer, one or more hidden layers, and one output layer. The external input is presented to the input layer, propagated forward through the hidden layers and creates the output vector in the output layer. Each unit i in the network computes its output with respect to its net input $net_i = \sum_j w_{ij} a_j$, where j represents all units in the previous layer that are connected to the unit i. The output of unit i is computed by passing the net input through a non-linear activation function f, i.e. $a_i = f(net_i)$.

The most commonly used non-linear activation functions are the log sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ or hyperbolic tangent sigmoid function $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$. The latter has been shown to be more suitable for binary classification problems.

The critical question is the computation of weights associated with the links connecting the neurons. In this paper, we use the resilient backpropagation (RPROP) algorithm (Riedmiller and Braun, 1993), which is based on the gradient descent method, but converges faster and generalizes better.

4 NeurAlign Approach

We propose a new approach, *NeurAlign*, that learns how to combine individual word alignment systems. We treat each alignment system as a classifier and transform the combination problem into a classifier ensemble problem. Before describing the NeurAlign approach, we first introduce some terminology used in the description below.

Let $E = e_1, \ldots, e_t$ and $F = f_1, \ldots, f_s$ be two sentences in two different languages. An alignment link (i, j) corresponds to a translational equivalence between words e_i and f_j. Let A_k be an alignment between sentences E and F, where each element $a \in A_k$ is an alignment link (i, j). Let $\mathcal{A} = \{A_1, \ldots, A_l\}$ be a set of alignments between E and F. We refer to the true alignment as T, where each $a \in T$ is of the form (i, j). A *neighborhood* of an alignment link (i, j)—denoted by $N(i, j)$—consists of 8 possible alignment links in a 3×3 window with (i, j) in the center of the window. Each element of $N(i, j)$ is called a *neighboring link* of (i, j).

Our goal is to combine the information in $A_1, \ldots, A_l$ such that the resulting alignment is closer to T. A straightforward solution is to take the intersection or union of the individual alignments, or perform a majority voting for each possible alignment link (i, j). Here, we use an additional model to learn how to combine outputs of $A_1, \ldots, A_l$.

We decompose the task of combining word alignments into two steps: (1) Extract features; and (2) Learn a classifier from the transformed data. We describe each of these two steps in turn.

4.1 Extracting Features

Given sentences E and F, we create a (potential) alignment instance (i, j) for all possible word combinations. A crucial component of building a classifier is the selection of features to represent the data. The simplest approach is to treat each alignment-system output as a separate feature upon which we build a classifier. However, when only a few alignment systems are combined, this feature space is not sufficient to distinguish between instances. One of the strategies in the classification literature is to supply the input data to the set of features as well.

While combining word alignments, we use two types of features to describe each instance (i, j): (1) linguistic features and (2) alignment features. Linguistic features include POS tags of both words (e_i and f_j) and a dependency relation for one of the words (e_i). We generate POS tags using the MXPOST tagger (Ratnaparkhi, 1996) for English and Chinese, and Connexor for Spanish. Dependency relations are produced using a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies.

Alignment features consist of features that are extracted from the outputs of individual alignment systems. For each alignment $A_k \in \mathcal{A}$, the following are some of the alignment features that can be used to describe an instance (i, j):

1. Whether (i, j) is an element of A_k or not
2. Translation probability $p(f_j|e_i)$ computed over A_k[1]
3. Fertility of (i.e., number of words in F that are aligned to) e_i in A_k
4. Fertility of (i.e., number of words in E that are aligned to) f_j in A_k
5. For each neighbor $(x, y) \in N(i, j)$, whether $(x, y) \in A_k$ or not (8 features in total)
6. For each neighbor $(x, y) \in N(i, j)$, translation probability $p(f_y|e_x)$ computed over A_k (8 features in total)

It is also possible to use variants, or combinations, of these features to reduce feature space.

Figure 2 shows an example of how we transform the outputs of 2 alignment systems, A_1 and A_2, for an alignment link (i, j) into data with some of the features above. We use -1 and 1 to represent the absence and existence of a link, respectively. The neighboring links are presented in row-by-row order.

A_1	f_{j-1}	f_j	f_{j+1}
e_{i-1}			
e_i	X	X	
e_{i+1}			X

A_2	f_{j-1}	f_j	f_{j+1}
e_{i-1}	X		
e_i			X
e_{i+1}			X

Features for the alignment link (*i* , *j*)	
pos(e_i) , pos(f_j)	Noun, Prep
rel(e_i)	Modifier
outputs of aligners	1 (for A_1), -1 (for A_2)
neighbors (A_1)	-1, -1, -1, **1**, -1, -1, -1, **1**
neighbors (A_2)	**1**, -1, -1, -1, **1**, -1, -1, **1**
neighbors ($A_1 \cup A_2$)	**1**, -1, -1, **1**, **1**, -1, -1, **1**
total neighbors	2 (for A_1), 3 (for A_2)
fertility(e_i)	2 (for A_1), 1 (for A_2)
fertility(f_j)	1 (for A_1), 0 (for A_2)

Figure 2: An Example of Transforming Alignments into Classification Data

For each sentence pair $E = e_1, \ldots, e_t$ and $F = f_1, \ldots, f_s$, we generate $s \times t$ instances to represent the sentence pair in the classification data.

Supervised learning requires the correct output, which here is the true alignment T. If an alignment link (i, j) is an element of T, then we set the correct output to 1, and to -1, otherwise.

4.2 Learning A Classifier

Once we transform the alignments into a set of instances with several features, the remaining task is to learn a classifier from this data. In the case of word alignment combination, there are important issues to consider for choosing an appropriate classifier. First, there is a very limited amount of manually annotated data. This may give rise to poor generalizations because it is very likely that unseen data include lots of cases that are not observed in the training data.

Second, the distribution of the data according to the classes is skewed. In a preliminary study on an English-Spanish data set, we found out that only 4% of the all word pairs are aligned to each other by humans, among a possible 158K word pairs. Moreover, only 60% of those aligned word pairs were

[1]The translation probabilities can be borrowed from the existing systems, if available. Otherwise, they can be generated from the outputs of individual alignment systems using likelihood estimates.

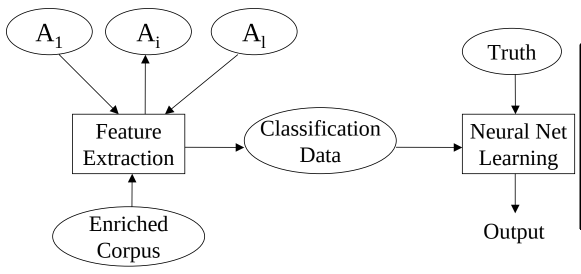


Figure 3: NeurAlign$_1$—Alignment Combination Using All Data At Once

		SPANISH						
		Adj	Adv	Comp	Det	Noun	Prep	Verb
E	Adj	18	-	-	82	40	96	66
N	Adv	-	8	-	-	50	67	75
G	Comp	-	-	12	-	46	37	96
L	Det	-	-	-	10	60	100	-
I	Noun	42	77	100	94	23	98	84
S	Prep	-	-	-	93	70	22	100
H	Verb	42	-	-	100	66	78	43

Table 1: Error Rates according to POS Tags for GIZA++ (E-to-S) (in percentages)

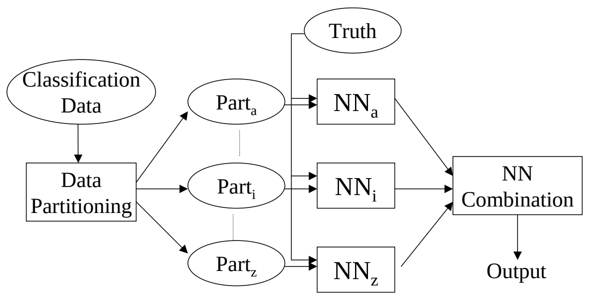


Figure 4: NeurAlign$_2$—Alignment Combination with Partitioning

also aligned by the individual alignment systems that were tested.

Finally, given the distribution of the data, it is difficult to find the right features to distinguish between instances. Thus, it is prudent to use as many features as possible and let the learning algorithm filter out the redundant features.

Below, we describe how neural nets are used at different levels to build a good classifier.

4.2.1 NeurAlign$_1$: Learning All At Once

Figure 3 illustrates how we combine alignments using all the training data at the same time (NeurAlign$_1$). First, the outputs of individual alignments systems and the original corpus (enriched with additional linguistic features) are passed to the feature extraction module. This module transforms the alignment problem into a classification problem by generating a training instance for every pair of words between the sentences in the original corpus. Each instance is represented by a set of features (described in Section 4.1). The new training data is passed to a neural net learner, which outputs whether an alignment link exists for each training instance.

4.2.2 NeurAlign$_2$: Multiple Neural Networks

The use of multiple neural networks (NeurAlign$_2$) enables the decomposition of a complex problem into smaller problems. *Local experts* are learned for each smaller problem and these are then merged. Following Tumer and Ghosh (1996), we apply spatial partitioning of training instances using proximity of patterns in the input space to reduce the complexity of the tasks assigned to individual classifiers.

We conducted a preliminary analysis on 100 randomly selected English-Spanish sentence pairs from a mixed corpus (UN + Bible + FBIS) to observe the distribution of errors according to POS tags in both languages. We examined the cases in which the individual alignment and the manual annotation were different—a total of 3,348 instances, where 1,320 of those are misclassified by GIZA++ (E-to-S).[2] We use a standard measure of error, i.e., the percentage of misclassified instances out of the total number of instances. Table 1 shows error rates (by percentage) according to POS tags for GIZA++ (E-to-S).[3]

Table 1 shows that the error rate is relatively low in cases where both words have the same POS tag. Except for verbs, the lowest error rate is obtained when both words have the same POS tag (the error rates on the diagonal). On the other hand, the error rates are high in several other cases, as much as 100%, e.g., when the Spanish word is a determiner or a preposition.[4] This suggests that dividing the training data according to POS tag, and training neural networks on each subset separately might be better than training on the entire data at once.

Figure 4 illustrates the combination approach with neural nets after partitioning the data into dis-

[2]For this analysis, we ignored the cases where both systems produced an output of -1 (i.e., the words are not aligned).

[3]Only POS pairs that occurred at least 10 times are shown.

[4]The same analysis was done for the other direction and resulted in similar distribution of error rates.

joint subsets (NeurAlign$_2$). Similar to NeurAlign$_1$, the outputs of individual alignment systems, as well as the original corpus, are passed to the feature extraction module. Then the training data is split into disjoint subsets using a subset of the available features for partitioning. We learn different neural nets for each partition, and then merge the outputs of the individual nets. The advantage of this is that it results in different generalizations for each partition and that it uses different subsets of the feature space for each net.

5 Experiments and Results

This section describes our experimental design, including evaluation metrics, data, and settings.

5.1 Evaluation Metrics

Let A be the set of alignment links for a set of sentences. We take S to be the set of sure alignment links and P be the set of probable alignment links (in the gold standard) for the same set of sentences. Precision (Pr), recall (Rc) and alignment error rate (AER) are defined as follows:

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

A manually aligned corpus is used as our gold standard. For English-Spanish data, the manual annotation is done by a bilingual English-Spanish speaker. Every link in the English-Spanish gold standard is considered a sure alignment link (i.e., $P = S$).

For English-Chinese, we used 2002 NIST MT evaluation test set. Each sentence pair was aligned by two native Chinese speakers, who are fluent in English. Each alignment link appearing in both annotations was considered a sure link, and links appearing in only one set were judged as probable. The annotators were not aware of the specifics of our approach.

5.2 Evaluation Data and Settings

We evaluated NeurAlign$_1$ and NeurAlign$_2$, using 5-fold cross validation on two data sets:

1. A set of 199 English-Spanish sentence pairs (nearly 5K words on each side) from a mixed corpus (UN + Bible + FBIS).
2. A set of 491 English-Chinese sentence pairs (nearly 13K words on each side) from 2002 NIST MT evaluation test set.

We computed precision, recall and error rate on the entire set of sentence pairs for each data set.[5]

To evaluate NeurAlign, we used GIZA++ in both directions (E-to-F and F-to-E, where F is either Chinese (C) or Spanish (S)) as input and a *refined alignment* approach (Och and Ney, 2000) that uses a heuristic combination method called *grow-diag-final* (Koehn et al., 2003) for comparison. (We henceforth refer to the refined-alignment approach as "RA.")

For the English-Spanish experiments, GIZA++ was trained on 48K sentence pairs from a mixed corpus (UN + Bible + FBIS), with nearly 1.2M of words on each side, using 10 iterations of Model 1, 5 iterations of HMM, and 5 iterations of Model 4. For the English-Chinese experiments, we used 107K sentence pairs from FBIS corpus (nearly 4.1M English and 3.3M Chinese words) to train GIZA++, using 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

5.3 Neural Network Settings

In our experiments, we used a multi-layer perceptron (MLP) consisting of 1 input layer, 1 hidden layer, and 1 output layer. The hidden layer consists of 10 units, and the output layer consists of 1 unit. All units in the hidden layer are fully connected to the units in the input layer, and the output unit is fully connected to all the units in the hidden layer. We used hyperbolic tangent sigmoid function as the activation function for both layers.

One of the potential pitfalls is overfitting as the number of iterations increases. To address this, we used the *early stopping with validation set* method. In our experiments, we held out (randomly selected) 1/4 of the training set as the validation set.

Neural nets are sensitive to the initial weights. To overcome this, we performed 5 runs of learning for each training set. The final output for each training is obtained by a majority voting over 5 runs.

[5]The number of alignment links varies over each fold. Therefore, we chose to evaluate all data at once instead of evaluating on each fold and then averaging.

5.4 Results

This section describes the experiments on English-Spanish and English-Chinese data for testing the effects of feature selection, training on the entire data (NeurAlign$_1$) or on the partitioned data (NeurAlign$_2$), using two input alignments: GIZA++ (E-to-F) and GIZA++ (F-to-E). We used the following additional features, as well as the outputs of individual aligners, for an instance (i, j) (set of features 2–7 below are generated separately for each input alignment A_k):

1. $posE_i, posF_j, relE_i$: POS tags and dependency relation for e_i and f_j.
2. $neigh(i, j)$: 8 features indicating whether a neighboring link exists in A_k.
3. $fertE_i, fertF_j$: 2 features indicating the fertility of e_i and f_j in A_k.
4. $NC(i, j)$: Total number of existing links in $N(i, j)$ in A_k.
5. $TP(i, j)$: Translation probability $p(f_j|e_i)$ in A_k.
6. $NghTP(i, j)$: 8 features indicating the translation probability $p(f_y|e_x)$ for each $(x, y) \in N(i, j)$ in A_k.
7. $AvTP(i, j)$: Average translation probability of the neighbors of (i, j) in A_k.

We performed statistical significance tests using two-tailed paired t-tests. Unless otherwise indicated, the differences between NeurAlign and other alignment systems, as well as the differences among NeurAlign variations themselves, were statistically significant within the 95% confidence interval.

5.4.1 Results for English-Spanish

Table 2 summarizes the precision, recall and alignment error rate values for each of our two alignment system inputs plus the three alternative alignment-combination approaches. Note that the best performing aligner among these is the RA method, with an AER of 21.2%. (We include this in subsequent tables for ease of comparison.)

Alignments	**Pr**	**Rc**	**AER**
E-to-S	87.0	67.0	24.3
S-to-E	88.0	67.5	23.6
Intersection	**98.2**	59.6	25.9
Union	80.6	**74.9**	22.3
RA	83.8	74.4	**21.2**

Table 2: Results for GIZA++ Alignments and Their Simple Combinations

Feature Selection for Training All Data At Once: NeurAlign$_1$ Table 3 presents the results of training neural nets using the entire data (NeurAlign$_1$) with different subsets of the feature space. When we used POS tags and the dependency relation as features, NeurAlign$_1$ performs worse than RA. Using the neighboring links as the feature set gave slightly (not significantly) better results than RA. Using POS tags, dependency relations, and neighboring links also resulted in better performance than RA but the difference was not statistically significant.

When we used fertilities along with the POS tags and dependency relations, the AER was 20.0%—a significant relative error reduction of 5.7% over RA. Adding the neighboring links to the previous feature set resulted in an AER of 17.6%—a significant relative error reduction of 17% over RA.

Interestingly, when we removed POS tags and dependency relations from this feature set, there was no significant change in the AER, which indicates that the improvement is mainly due to the neighboring links. This supports our initial claim about the clustering of alignment links, i.e., when there is an alignment link, usually there is another link in its neighborhood. Finally, we tested the effects of using translation probabilities as part of the feature set, and found out that using translation probabilities did no better than the case where they were not used. We believe this happens because the translation probability $p(f_j|e_i)$ has a unique value for each pair of e_i and f_j; therefore it is not useful to distinguish between alignment links with the same words.

Feature Selection for Training on Partitioned Data: NeurAlign$_2$ In order to train on partitioned data (NeurAlign$_2$), we needed to establish appropriate features for partitioning the training data. Table 4 presents the evaluation results for NeurAlign$_1$ (i.e., no partitioning) and NeurAlign$_2$ with different features for partitioning (English POS tag, Spanish POS tag, and POS tags on both sides). For training on each partition, the feature space included POS tags (e.g., Spanish POS tag in the case where partitioning is based on English POS tag only), dependency relations, neighborhood features, and fertilities. We observed that partitioning based on POS tags on one side reduced the AER to 17.4% and

Features	**Pr**	**Rc**	**AER**
$posE_i, posF_j, relE_i$	90.6	67.7	22.5
$neigh(i,j)$	91.3	69.5	21.1
$posE_i, posF_j, relE_i,$ $neigh(i,j)$	**91.7**	70.2	20.5
$posE_i, posF_j, relE_i,$ $fertE_i, fertF_j$	91.4	71.1	20.0
$posE_i, posF_j, relE_i,$ $neigh(i,j), NC(i,j)$ $fertE_i, fertF_j$	89.5	**76.3**	**17.6**
$neigh(i,j), NC(i,j)$ $fertE_i, fertF_j$	89.7	75.7	17.9
$posE_i, posF_j, relE_i,$ $fertE_i, fertF_j,$ $neigh(i,j), NC(i,j),$ $TP(i,j), AvTP(i,j)$	90.0	75.7	17.9
RA	83.8	74.4	21.2

Table 3: Combination with Neural Networks: NeurAlign$_1$ (All-Data-At-Once)

17.1%, respectively. Using POS tags on *both* sides reduced the error rate to 16.9%—a significant relative error reduction of 5.6% over no partitioning. All four methods yielded statistically significant error reductions over RA—we will examine the fourth method in more detail below.

Alignment	**Pr**	**Rc**	**AER**
NeurAlign$_1$	89.7	75.7	17.9
NeurAlign$_2[posE_i]$	91.1	75.4	17.4
NeurAlign$_2[posF_j]$	91.2	**76.0**	17.1
NeurAlign$_2[posE_i, posF_j]$	**91.6**	**76.0**	**16.9**
RA	83.8	74.4	21.2

Table 4: Effects of Feature Selection for Partitioning

Once we determined that partitioning by POS tags on both sides brought about the biggest gain, we ran NeurAlign$_2$ using this partitioning, but with different feature sets. Table 5 shows the results of this experiment. Using dependency relations, word fertilities and translation probabilities (both for the link in question and the neighboring links) yielded a significantly lower AER (18.6%)—a relative error reduction of 12.3% over RA. When the feature set consisted of dependency relations, word fertilities, and neighborhood links, the AER was reduced to 16.9%—a 20.3% relative error reduction over RA. We also tested the effects of adding translation probabilities to this feature set, but as in the case of NeurAlign$_1$, this did not improve the alignments.

In the best case, NeurAlign$_2$ achieved substantial and significant reductions in AER over the input alignment systems: a 28.4% relative error reduction over S-to-E and a 30.5% relative error reduction over E-to-S. Compared to RA, NeurAlign$_2$ also achieved significantly better results over RA: relative improvements of 9.3% in precision, 2.2% in recall, and 20.3% in AER.

Features	**Pr**	**Rc**	**AER**
$relE_i, fertE_i, fertF_j,$ $TP(i,j), AvTP(i,j),$ $NghTP(i,j)$	**91.9**	73.0	18.6
$neigh(i,j)$	90.3	74.0	18.7
$relE_i, fertE_i, fertF_j,$ $neigh(i,j), NC(i,j)$	91.6	76.0	**16.9**
$relE_i, fertE_i, fertF_j,$ $neigh(i,j), NC(i,j),$ $TP(i,j), AvTP(i,j)$	91.4	**76.1**	**16.9**
RA	83.8	74.4	21.2

Table 5: Combination with Neural Networks: NeurAlign$_2$ (Partitioned According to POS tags)

5.4.2 Results for English-Chinese

The results of the input alignments to NeurAlign, i.e., GIZA++ alignments in two different directions, NeurAlign$_1$ (i.e., no partitioning) and variations of NeurAlign$_2$ with different features for partitioning (English POS tag, Chinese POS tag, and POS tags on both sides) are shown in Table 6. For comparison, we also include the results for RA in the table. For brevity, we include only the features resulting in the best configurations from the English-Spanish experiments, i.e., POS tags, dependency relations, word fertilities, and neighborhood links (the features in the third row of Table 5). The ground truth used during the training phase consisted of all the alignment links with equal weight.

Alignments	**Pr**	**Rc**	**AER**
E-to-C	70.4	68.3	30.7
C-to-E	66.0	69.8	32.2
NeurAlign$_1$	85.0	71.4	22.2
NeurAlign$_2[posE_i]$	85.7	74.6	20.0
NeurAlign$_2[posF_j]$	85.7	73.2	20.8
NeurAlign$_2[posE_i, posF_j]$	**86.3**	74.7	**19.7**
RA	61.9	**82.6**	29.7

Table 6: Results on English-Chinese Data

Without any partitioning, NeurAlign achieves an alignment error rate of 22.2%—a significant relative error reduction of 25.3% over RA. Partitioning the data according to POS tags results in significantly better results over no partitioning. When the data is partitioned according to both POS tags, NeurAlign reduces AER to 19.7%—a significant relative error reduction of 33.7% over RA. Compared to the input alignments, the best version of NeurAlign achieves a relative error reduction of 35.8% and 38.8%, respectively.

6 Conclusions

We presented NeurAlign, a novel approach to combining the outputs of different word alignment systems. Our approach treats individual alignment systems as black boxes, and transforms the individual alignments into a set of data with features that are borrowed from their outputs and additional linguistic features (such as POS tags and dependency relations). We use neural nets to learn the true alignments from these transformed data.

We show that using POS tags to partition the transformed data, and learning a different classifier for each partition is more effective than using the entire data at once. Our results indicate that NeurAlign yields a significant 28-39% relative error reduction over the best of the input alignment systems and a significant 20-34% relative error reduction over the best known alignment combination technique on English-Spanish and English-Chinese data.

We should note that NeurAlign is not a standalone word alignment system but a supervised learning approach to improve already existing alignment systems. A drawback of our approach is that it requires annotated data. However, our experiments have shown that significant improvements can be obtained using a small set of annotated data. We will do additional experiments to observe the effects of varying the size of the annotated data while learning neural nets. We are also planning to investigate whether NeurAlign helps when the individual aligners are trained using more data.

We will extend our combination approach to combine word alignment systems based on different models, and investigate the effectiveness of our technique on other language pairs. We also intend to evaluate the effectiveness of our improved alignment approach in the context of machine translation and cross-language projection of resources.

Acknowledgments This work has been supported in part by ONR MURI Contract FCPO.810548265, Cooperative Agreement DAAD190320020, and NSF ITR Grant IIS-0326553.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and PP attachment. In *Proceedings of EMNLP'1999*, pages 38–45.

Necip F. Ayan, Bonnie J. Dorr, and Nizar Habash. 2004. Multi-Align: Combining linguistic and statistical techniques to improve alignments for adaptable MT. In *Proceedings of AMTA'2004*, pages 17–26.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proc. of ACL'1998*.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL'2003*.

Micheal Collins. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of ACL'1997*.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of ACL'2002*.

Bonnie J. Dorr, Lisa Pearl, Rebecca Hwa, and Nizar Habash. 2002. DUSTer: A method for unraveling cross-language divergences for statistical word–level alignment. In *Proceedings of AMTA'2002*.

Radu Florian and David Yarowsky. 2002. Modeling consensus: Classifier combination for word sense disambiguation. In *Proceedings of EMNLP'2002*, pages 25–32.

L. Hansen and P. Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

John C. Henderson and Eric Brill. 2000. Bagging and boosting a treebank parser. In *Proceedings of NAACL'2000*.

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

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL'2005*.

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

Marvin Minsky. 1999. Logical Versus Analogical or Symbolic Versus Connectionist or Neat Versus Scruffy. *AI Magazine*, 12:34–51.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL'2000*.

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

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP'1996*.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE Intl. Conf. on Neural Networks*, pages 586–591.

Jorg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of EACL'2003*, pages 339–346.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of EMNLP'2002*.

Kagan Tumer and Joydeep Ghosh. 1996. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3–4):385–404, December.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT'2001*.

A Discriminative Matching Approach to Word Alignment

Ben Taskar   Simon Lacoste-Julien   Dan Klein
Computer Science Division, EECS Department
University of California, Berkeley
Berkeley, CA 94720

Abstract

We present a discriminative, large-margin approach to feature-based matching for word alignment. In this framework, pairs of word tokens receive a matching score, which is based on features of that pair, including measures of association between the words, distortion between their positions, similarity of the orthographic form, and so on. Even with only 100 labeled training examples and simple features which incorporate counts from a large unlabeled corpus, we achieve AER performance close to IBM Model 4, in much less time. Including Model 4 predictions as features, we achieve a relative AER reduction of 22% in over intersected Model 4 alignments.

1 Introduction

The standard approach to word alignment from sentence-aligned bitexts has been to construct models which generate sentences of one language from the other, then fitting those generative models with EM (Brown et al., 1990; Och and Ney, 2003). This approach has two primary advantages and two primary drawbacks. In its favor, generative models of alignment are well-suited for use in a noisy-channel translation system. In addition, they can be trained in an unsupervised fashion, though in practice they do require labeled validation alignments for tuning model hyper-parameters, such as null counts or smoothing amounts, which are crucial to producing alignments of good quality. A primary drawback of the generative approach to alignment is that, as in all generative models, explicitly incorporating arbitrary features of the input is difficult. For example, when considering whether to align two words in the IBM models (Brown et al., 1990), one cannot easily include information about such features as orthographic similarity (for detecting cognates), presence of the pair in various dictionaries, similarity of the frequency of the two words, choices made by other alignment systems on this sentence pair, and so on. While clever models can implicitly capture some of these information sources, it takes considerable work, and can make the resulting models quite complex. A second drawback of generative translation models is that, since they are learned with EM, they require extensive processing of large amounts of data to achieve good performance. While tools like GIZA++ (Och and Ney, 2003) do make it easier to build on the long history of the generative IBM approach, they also underscore how complex high-performance generative models can, and have, become.

In this paper, we present a discriminative approach to word alignment. Word alignment is cast as a maximum weighted matching problem (Cormen et al., 1990) in which each pair of words (e_j, f_k) in a sentence pair (e, f) is associated with a score $s_{jk}(e, f)$ reflecting the desirability of the alignment of that pair. The alignment for the sentence pair is then the highest scoring matching under some constraints, for example the requirement that matchings be one-to-one.

This view of alignment as graph matching is not, in itself, new: Melamed (2000) uses competitive linking to greedily construct matchings where the pair score is a measure of word-to-word association, and Matusov et al. (2004) find exact maximum matchings where the pair scores come from the alignment posteriors of generative models. Tiedemann (2003) proposes incorporating a variety of word association "clues" into a greedy linking algorithm.

What we contribute here is a principled approach for tractable and efficient learning of the alignment score $s_{jk}(e, f)$ as a function of arbitrary features of that token pair. This contribution opens up the possibility of doing the kind of feature engineering for alignment that has been so successful for other NLP tasks. We first present the algorithm for large margin estimation of the scoring function. We then show that our method can achieve AER rates comparable to unsymmetrized IBM Model 4, using extremely little labeled data (as few as 100 sentences) and a simple feature set. Remarkably, by including bi-directional IBM Model 4 predictions as features, we achieve an absolute AER of 5.4 on the English-French Hansards alignment task, a relative reduction of 22% in AER over intersected Model 4 alignments and, to our knowledge, the best AER result published on this task.

2 Algorithm

We model the alignment prediction task as a maximum weight bipartite matching problem, where nodes correspond to the words in the two sentences. For simplicity, we assume here that each word aligns to one or zero words in the other sentence. The edge weight s_{jk} represents the degree to which word j in one sentence can translate into the word k in the other sentence. Our goal is to find an alignment that maximizes the sum of edge scores. We represent a matching using a set of binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk} s_{jk} y_{jk}$. The maximum weight bipartite matching problem, $\arg\max_{\mathbf{y}\in\mathcal{Y}} s(\mathbf{y})$, can be solved using well known combinatorial algorithms or the following linear program:

$$\begin{aligned} \max_{\mathbf{z}} \quad & \sum_{jk} s_{jk} z_{jk} && (1) \\ \text{s.t.} \quad & \sum_{j} z_{jk} \le 1, \quad \sum_{k} z_{jk} \le 1, \quad 0 \le z_{jk} \le 1, \end{aligned}$$

where the continuous variables z_{jk} correspond to the binary variables y_{jk}. This LP is guaranteed to have integral (and hence optimal) solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003). Note that although the above LP can be used to compute alignments, combinatorial algorithms are generally more efficient. However, we use the LP to develop the learning algorithm below.

For a sentence pair $\mathbf{x}$, we denote position pairs by $\mathbf{x}_{jk}$ and their scores as s_{jk}. We let $s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$ for some user provided feature mapping $\mathbf{f}$ and abbreviate $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$. We can include in the feature vector the identity of the two words, their relative positions in their respective sentences, their part-of-speech tags, their string similarity (for detecting cognates), and so on.

At this point, one can imagine estimating a linear matching model in multiple ways, including using conditional likelihood estimation, an averaged perceptron update (see which matchings are proposed and adjust the weights according to the difference between the guessed and target structures (Collins, 2002)), or in large-margin fashion. Conditional likelihood estimation using a log-linear model $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$ requires summing over all matchings to compute the normalization $Z_{\mathbf{w}}(\mathbf{x})$, which is #P-complete (Valiant, 1979). In our experiments, we therefore investigated the averaged perceptron in addition to the large-margin method outlined below.

2.1 Large-margin estimation

We follow the large-margin formulation of Taskar et al. (2005a). Our input is a set of training instances $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{m}$, where each instance consists of a sentence pair $\mathbf{x}_i$ and a target alignment $\mathbf{y}_i$. We would like to find parameters $\mathbf{w}$ that predict correct alignments on the training data:

$$\mathbf{y}_i = \arg\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \bar{\mathbf{y}}_i), \quad \forall i,$$

where $\mathcal{Y}_i$ is the space of matchings appropriate for the sentence pair i.

In standard classification problems, we typically measure the error of prediction, $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, using the simple 0-1 loss. In structured problems, where we are jointly predicting multiple variables, the loss is often more complex. While the F-measure is a natural loss function for this task, we instead chose a sensible surrogate that fits better in our framework: Hamming distance between $\mathbf{y}_i$ and $\bar{\mathbf{y}}_i$, which simply counts the number of edges predicted incorrectly.

We use an SVM-like hinge upper bound on the loss $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, given by $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i}[\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)]$, where $\ell_i(\bar{\mathbf{y}}_i) = \ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, and $\mathbf{f}_i(\bar{\mathbf{y}}_i) = \mathbf{f}(\mathbf{x}_i, \bar{\mathbf{y}}_i)$. Minimizing this upper bound encourages the true alignment $\mathbf{y}_i$ to be optimal with respect to $\mathbf{w}$ for each instance i:

$$\min_{||\mathbf{w}|| \le \gamma} \sum_i \max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i),$$

where γ is a regularization parameter.

In this form, the estimation problem is a mixture of continuous optimization over $\mathbf{w}$ and combinatorial optimization over $\mathbf{y}_i$. In order to transform it into a more standard optimization problem, we need a way to efficiently handle the *loss-augmented inference,* $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i}[\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i)]$. This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn — $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i)$ — but with an additional term corresponding to the loss function. Our assumption that the loss function decomposes over the edges is crucial to solving this problem. In particular, we use weighted Hamming distance, which counts the number of variables in which a candidate solution $\bar{\mathbf{y}}_i$ differs from the target output $\mathbf{y}_i$, with different cost for false positives (c^+) and false negatives (c^-):

$$\ell_i(\bar{\mathbf{y}}_i) = \sum_{jk} [c^- y_{i,jk}(1 - \bar{y}_{i,jk}) + c^+ \bar{y}_{i,jk}(1 - y_{i,jk})]$$

$$= \sum_{jk} c^- y_{i,jk} + \sum_{jk} [c^+ - (c^- + c^+) y_{i,jk}] \bar{y}_{i,jk}.$$

The loss-augmented matching problem can then be written as an LP similar to Equation 1 (without the constant term $\sum_{jk} c^- y_{i,jk}$):

$$\max_{\mathbf{z}} \quad \sum_{jk} z_{i,jk} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{i,jk}) + c^+ - (c^- + c^+) y_{i,jk}]$$

$$\text{s.t.} \quad \sum_j z_{i,jk} \le 1, \quad \sum_k z_{i,jk} \le 1, \quad 0 \le z_{i,jk} \le 1.$$

Hence, without any approximations, we have a continuous optimization problem instead of a combinatorial one:

$$\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i) = d_i + \max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{w}^\top \mathbf{F}_i + \mathbf{c}_i)^\top \mathbf{z}_i,$$

where $d_i = \sum_{jk} c^- y_{i,jk}$ is the constant term, $\mathbf{F}_i$ is the appropriate matrix that has a column of features $\mathbf{f}(\mathbf{x}_{i,jk})$ for each edge jk, $\mathbf{c}_i$ is the vector of the loss terms $c^+ - (c^- + c^+) y_{i,jk}$ and finally $\mathcal{Z}_i = \{\mathbf{z}_i : \sum_j z_{i,jk} \le 1, \ \sum_k z_{i,jk} \le 1, \ 0 \le z_{i,jk} \le 1\}$.

Plugging this LP back into our estimation problem, we have

$$\min_{||\mathbf{w}|| \le \gamma} \max_{\mathbf{z} \in \mathcal{Z}} \sum_i \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{F}_i \mathbf{y}_i, \quad (2)$$

where $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$, $\mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_m$. Instead of the derivation in Taskar et al. (2005a), which produces a joint convex optimization problem using Lagrangian duality, here we tackle the problem in its natural saddle-point form.

2.2 The extragradient method

For saddle-point problems, a well-known solution strategy is the extragradient method (Korpelevich, 1976), which is closely related to projected-gradient methods.

The gradient of the objective in Equation 2 is given by: $\sum_i \mathbf{F}_i(\mathbf{z}_i - \mathbf{y}_i)$ (with respect to $\mathbf{w}$) and $\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i$ (with respect to each $\mathbf{z}_i$). We denote the Euclidean projection of a vector onto $\mathcal{Z}_i$ as $P_{\mathcal{Z}_i}(\mathbf{v}) = \arg\min_{\mathbf{u} \in \mathcal{Z}_i} ||\mathbf{v} - \mathbf{u}||$ and projection onto the ball $||\mathbf{w}|| \le \gamma$ as $P_\gamma(\mathbf{w}) = \gamma \mathbf{w} / \max(\gamma, ||\mathbf{w}||)$.

An iteration of the extragradient method consists of two very simple steps, prediction:

$$\bar{\mathbf{w}}^{t+1} = P_\gamma(\mathbf{w}^t + \beta_k \sum_i \mathbf{F}_i(\mathbf{y}_i - \mathbf{z}_i^t));$$
$$\bar{\mathbf{z}}_i^{t+1} = P_{\mathcal{Z}_i}(\mathbf{z}_i^t + \beta_k(\mathbf{F}_i^\top \mathbf{w}^t + \mathbf{c}_i));$$

and correction:

$$\mathbf{w}^{t+1} = P_\gamma(\mathbf{w}^t + \beta_k \sum_i \mathbf{F}_i(\mathbf{y}_i - \bar{\mathbf{z}}_i^{t+1}));$$
$$\mathbf{z}_i^{t+1} = P_{\mathcal{Z}_i}(\mathbf{z}_i^t + \beta_k(\mathbf{F}_i^\top \bar{\mathbf{w}}^{t+1} + \mathbf{c}_i)),$$

where β_k are appropriately chosen step sizes. The method is guaranteed to converge linearly to a solution $\mathbf{w}^*, \mathbf{z}^*$ (Korpelevich, 1976; He and Liao, 2002; Taskar et al., 2005b). Please see www.cs.berkeley.edu/~taskar/extragradient.pdf for more details.

The key subroutine of the algorithm is Euclidean projection onto the feasible sets $\mathcal{Z}_i$. In case of word alignment, $\mathcal{Z}_i$ is the convex hull of bipartite matchings and the problem reduces to the much-studied minimum cost quadratic flow problem (Bertsekas et al., 1997). The projection problem $P_{\mathcal{Z}_i}(\mathbf{z}'_i)$ is given by

$$\begin{aligned} \min_{\mathbf{z}} \quad & \sum_{jk} \frac{1}{2}(z'_{i,jk} - z_{i,jk})^2 \\ \text{s.t.} \quad & \sum_j z_{i,jk} \leq 1, \quad \sum_k z_{i,jk} \leq 1, \quad 0 \leq z_{i,jk} \leq 1. \end{aligned}$$

We can now use a standard reduction of bipartite matching to min cost flow by introducing a source node connected to all the words in one sentence and a sink node connected to all the words in the other sentence, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{i,jk} - z_{i,jk})^2$ and capacity 1. Now the minimum cost flow from the source to the sink computes projection of $\mathbf{z}'_i$ onto $\mathcal{Z}_i$ We use standard, publicly-available code for solving this problem (Guerriero and Tseng, 2002).

3 Experiments

We applied this matching algorithm to word-level alignment using the English-French Hansards data from the 2003 NAACL shared task (Mihalcea and Pedersen, 2003). This corpus consists of 1.1M automatically aligned sentences, and comes with a validation set of 39 sentence pairs and a test set of 447 sentences. The validation and test sentences have been hand-aligned (see Och and Ney (2003)) and are marked with both *sure* and *possible* alignments. Using these alignments, *alignment error rate* (AER) is calculated as:

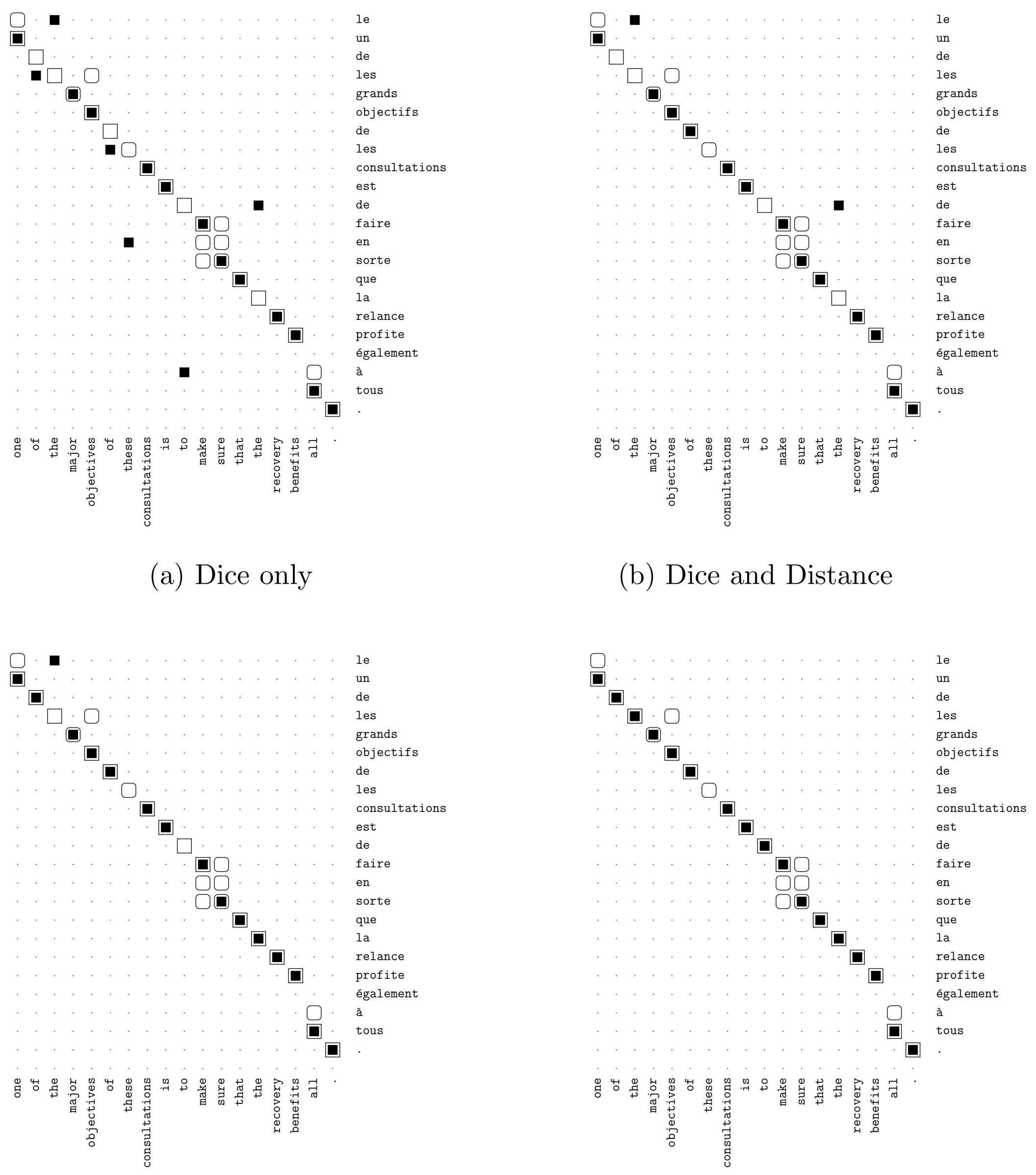
$$AER(A, S, P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

Here, A is a set of proposed index pairs, S is the sure gold pairs, and P is the possible gold pairs. For example, in Figure 1, proposed alignments are shown against gold alignments, with open squares for sure alignments, rounded open squares for possible alignments, and filled black squares for proposed alignments.

Since our method is a supervised algorithm, we need labeled examples. For the training data, we split the original test set into 100 training examples and 347 test examples. In all our experiments, we used a structured loss function $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$ that penalized false negatives 3 times more than false positives, where 3 was picked by testing several values on the validation set. Instead of selecting a regularization parameter γ and running to convergence, we used early stopping as a cheap regularization method, by setting γ to a very large value (10000) and running the algorithm for 500 iterations. We selected a stopping point using the validation set by simply picking the best iteration on the validation set in terms of AER (ignoring the initial ten iterations, which were very noisy in our experiments). All selected iterations turned out to be in the first 50 iterations, as the algorithm converged fairly rapidly.

3.1 Features and Results

Very broadly speaking, the classic IBM models of word-level translation exploit four primary sources of knowledge and constraint: association of words (all IBM models), competition between alignments (all models), zero- or first-order preferences of alignment positions (2,4+), and fertility (3+). We model all of these in some way,



(a) Dice only (b) Dice and Distance

(c) Dice, Distance, Orthographic, and BothShort (d) All features

Figure 1: Example alignments for each successive feature set.

except fertility.[1]

First, and, most importantly, we want to include information about word association; translation pairs are likely to co-occur together in a bitext. This information can be captured, among many other ways, using a feature whose value is the Dice coefficient (Dice, 1945):

$$Dice(e,f) = \frac{2C_{EF}(e,f)}{C_E(e)C_F(f)}$$

Here, C_E and C_F are counts of word occurrences in each language, while C_{EF} is the number of co-occurrences of the two words. With just this feature on a pair of word tokens (which depends only on their types), we can already make a stab

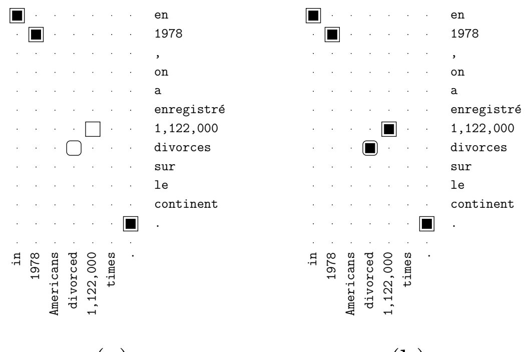
[1]In principle, we can model also model fertility, by allowing 0-k matches for each word rather than 0-1, and having bias features on each word. However, we did not explore this possibility.

at word alignment, aligning, say, each English word with the French word (or null) with the highest Dice value (see (Melamed, 2000)), simply as a matching-free heuristic model. With Dice counts taken from the 1.1M sentences, this gives and AER of 38.7 with English as the target, and 36.0 with French as the target (in line with the numbers from Och and Ney (2003)).

As observed in Melamed (2000), this use of Dice misses the crucial constraint of competition: a candidate source word with high association to a target word may be unavailable for alignment because some other target has an even better affinity for that source word. Melamed uses competitive linking to incorporate this constraint explicitly, while the IBM-style models get this effect via explaining-away effects in EM training. We can get something much like the combination of Dice and competitive linking by running with just one feature on each pair: the Dice value of that pair's words.[2] With just a Dice feature – meaning no learning is needed yet – we achieve an AER of 29.8, between the Dice with competitive linking result of 34.0 and Model 1 of 25.9 given in Och and Ney (2003). An example of the alignment at this stage is shown in Figure 1(a). Note that most errors lie off the diagonal, for example the often-correct *to-à* match.

IBM Model 2, as usually implemented, adds the preference of alignments to lie near the diagonal. Model 2 is driven by the product of a word-to-word measure and a (usually) Gaussian distribution which penalizes distortion from the diagonal. We can capture the same effect using features which reference the relative positions j and k of a pair (e_j, f_k). In addition to a Model 2-style quadratic feature referencing relative position, we threw in the following proximity features: absolute difference in relative position $abs(j/|e|-k/|f|)$, and the square and square root of this value. In addition, we used a conjunction feature of the dice coefficient times the proximity. Finally, we added a bias feature on each edge, which acts as a threshold that allows

[2]This isn't quite competitive linking, because we use a non-greedy matching.



(a) (b)

Figure 2: Example alignments showing the effects of orthographic cognate features. (a) Dice and Distance, (b) With Orthographic Features.

sparser, higher precision alignments. With these features, we got an AER of 15.5 (compare to 19.5 for Model 2 in (Och and Ney, 2003)). Note that we already have a capacity that Model 2 does not: we can learn a non-quadratic penalty with linear mixtures of our various components – this gives a similar effect to learning the variance of the Gaussian for Model 2, but is, at least in principle, more flexible.[3] These features fix the *to-à* error in Figure 1(a), giving the alignment in Figure 1(b).

On top of these features, we included other kinds of information, such as word-similarity features designed to capture cognate (and exact match) information. We added a feature for exact match of words, exact match ignoring accents, exact matching ignoring vowels, and fraction overlap of the longest common subsequence. Since these measures were only useful for long words, we also added a feature which indicates that both words in a pair are short. These orthographic and other features improved AER to 14.4. The running example now has the alignment in Figure 1(c), where one improvement may be attributable to the short pair feature – it has stopped proposing *the-de*, partially because the short pair feature downweights the score of that pair. A clearer example of these features making a difference is shown in Figure 2, where both the exact-match and character overlap fea-

[3]The learned response was in fact close to a Gaussian, but harsher near zero displacement.

tures are used.

One source of constraint which our model still does not explicitly capture is the first-order dependency between alignment positions, as in the HMM model (Vogel et al., 1996) and IBM models 4+. The *the-le* error in Figure 1(c) is symptomatic of this lack. In particular, it is a slightly better pair according to the Dice value than the correct *the-les*. However, the latter alignment has the advantage that *major-grands* follows it. To use this information source, we included a feature which gives the Dice value of the words following the pair.[4] We also added a word-frequency feature whose value is the absolute difference in log rank of the words, discouraging very common words from translating to very rare ones. Finally, we threw in bilexical features of the pairs of top 5 non-punctuation words in each language.[5] This helped by removing specific common errors like the residual tendency for French *de* to mistakenly align to English *the* (the two most common words). The resulting model produces the alignment in Figure 1(d). It has sorted out the *the-le* / *the-les* confusion, and is also able to guess *to-de*, which is not the most common translation for either word, but which is supported by the good Dice value on the following pair (*make-faire*).

With all these features, we got a final AER of 10.7, broadly similar to the 8.9 or 9.7 AERs of unsymmetrized IBM Model 4 trained on the same data that the Dice counts were taken from.[6] Of course, symmetrizing Model 4 by intersecting alignments from both directions does yield an improved AER of 6.9, so, while our model does do surprisingly well with cheaply obtained count-based features, Model 4 does still outperform it so far. However, our model can also easily incorporate the predictions of Model 4 as additional features. We therefore added three new features for each edge: the prediction of Model 4 in the English-French direction, the prediction in the French-English direction, and the intersection of the two predictions. With these powerful new features, our AER dropped dramatically to 5.4, a 22% improvement over the intersected Model 4 performance.

Model	AER
Dice (without matching)	38.7 / 36.0
Model 4 (E-F, F-E, intersected)	8.9 / 9.7/ 6.9
Discriminative Matching	
Dice Feature Only	29.8
+ Distance Features	15.5
+ Word Shape and Frequency	14.4
+ Common Words and Next-Dice	10.7
+ Model 4 Predictions	5.4

Figure 3: AER on the Hansards task.

Another way of doing the parameter estimation for this matching task would have been to use an averaged perceptron method, as in Collins (2002). In this method, we merely run our matching algorithm and update weights based on the difference between the predicted and target matchings. However, the performance of the average perceptron learner on the same feature set is much lower, only 8.1, not even breaking the AER of its best single feature (the intersected Model 4 predictions).

3.2 Scaling Experiments

We explored the scaling of our method by learning on a larger training set, which we created by using GIZA++ intersected bi-directional Model 4 alignments for the unlabeled sentence pairs. We then took the first 5K sentence pairs from these 1.1M Model 4 alignments. This gave us more training data, albeit with noisier labels. On a 3.4GHz Intel Xeon CPU, GIZA++ took 18 hours to align the 1.1M words, while our method learned its weights in between 6 minutes (100 training sentences) and three hours (5K sentences).

[4]It is important to note that while our matching algorithm has no first-order effects, the features can encode such effects in this way, or in better ways – e.g. using as features posteriors from the HMM model in the style of Matusov et al. (2004).

[5]The number of such features which can be learned depends on the number of training examples, and since some of our experiments used only a few dozen training examples we did not make heavy use of this feature.

[6]Note that the common word pair features affected common errors and therefore had a particularly large impact on AER.

4 Conclusions

We have presented a novel discriminative, large-margin method for learning word-alignment models on the basis of arbitrary features of word pairs. We have shown that our method is suitable for the common situation where a moderate number of good, fairly general features must be balanced on the basis of a small amount of labeled data. It is also likely that the method will be useful in conjunction with a large labeled alignment corpus (should such a set be created). We presented features capturing a few separate sources of information, producing alignments on the order of those given by unsymmetrized IBM Model 4 (using labeled training data of about the size others have used to tune generative models). In addition, when given bi-directional Model 4 predictions as features, our method provides a 22% AER reduction over intersected Model 4 predictions alone. The resulting 5.4 AER on the English-French Hansarks task is, to our knowledge, the best published AER figure for this training scenario (though since we use a subset of the test set, evaluations are not problem-free). Finally, our method scales to large numbers of training sentences and trains in minutes rather than hours or days for the higher-numbered IBM models, a particular advantage when not using features derived from those slower models.

References

D. P. Bertsekas, L. C. Polymenakos, and P. Tseng. 1997. An e-relaxation method for separable convex cost network flow problems. *SIAM J. Optim.*, 7(3):853–870.

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to Algorithms.* MIT Press, Cambridge, MA.

L. R. Dice. 1945. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302.

F. Guerriero and P. Tseng. 2002. Implementation and test of auction methods for solving generalized network flow problems with separable convex cost. *Journal of Optimization Theory and Applications*, 115(1):113–144, October.

B.S. He and L. Z. Liao. 2002. Improvements of some projection methods for monotone nonlinear variational inequalities. *JOTA*, 112:111:128.

G. M. Korpelevich. 1976. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747:756.

E. Matusov, R. Zens, and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proc. of COLING 2004*.

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

R. Mihalcea and T. 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*, pages 1–6, Edmonton, Alberta, Canada.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. 2005a. Learning structured prediction models: a large margin approach. In *Proceedings of the International Conference on Machine Learning.*

B. Taskar, S. Lacoste-Julien, and M. Jordan. 2005b. Structured prediction via the extragradient method. In *Proceedings of Neural Information Processing Systems.*

J. Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of EACL.*

L. G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING 16*, pages 836–841.

A Discriminative Framework for Bilingual Word Alignment

Robert C. Moore
Microsoft Research
One Microsoft Way
Redmond, WA 98052
bobmoore@microsoft.com

Abstract

Bilingual word alignment forms the foundation of most approaches to statistical machine translation. Current word alignment methods are predominantly based on generative models. In this paper, we demonstrate a discriminative approach to training simple word alignment models that are comparable in accuracy to the more complex generative models normally used. These models have the the advantages that they are easy to add features to and they allow fast optimization of model parameters using small amounts of annotated data.

1 Motivation

Bilingual word alignment is the first step of most current approaches to statistical machine translation. Although the best performing systems are “phrase-based” (e.g, Och and Ney, 2004), possible phrase translations are normally first extracted from word-aligned bilingual text segments. The standard approach to word alignment makes use of various combinations of five generative 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). The best combinations of these models can produce high accuracy alignments, at least when trained on a large corpus of fairly direct translations in related languages.

These standard models are less than ideal, however, in a number of ways, two of which we address in this paper. First, although the standard models can theoretically be trained without supervision, in practice various parameters are introduced that should be optimized using annotated data. For, example, Och and Ney (2003) suggest supervised optimization of a number of parameters, including the probablity of jumping to the empty word in the HMM model, as well as smoothing parameters for the distortion probabilities and fertility probabilities of the more complex models. Since the values of these parameters affect the values of the translation, alignment, and fertility probabilities trained by EM, there is no effective way to optimize them other than to run the training procedure with a particular combination of values and evaluate the accuracy of the resulting alignments. Since evaluating each combination of parameter values in this way can take hours to days on a large training corpus, it seems safe to say that these parameters are rarely if ever truly jointly optimized for a particular alignment task.

The second problem we address is the difficulty of adding features to the standard generative models. Generative models require a generative “story” as to how the observed data is generated by an interrelated set of stochastic processes. For example, the generative story for IBM Models 1 and 2 and the HMM alignment model is that a target language translation of a given source language sentence is generated by first choosing a length for the target language sentence, then for each target sentence position choosing a source sentence word, and then choosing the corresponding target language word. When Brown et al. (1993) wanted to add a fertility component to create Models 3, 4, and 5, however, this generative story didn't fit any longer, because it does not include how many target language words to align to each source language word as a separate decision. To model this explicitly, they had to come up with a different generative story.

In this paper, we take a different approach to word alignment, based on discriminative training of a weighted linear combination of a small number of features. For a given parallel sentence pair, for each possible word alignment considered, we simply multiply the values of each of these features by a corresponding weight to give a score for that feature, and sum the features scores to give an overall score for the alignment. The possible alignment having the best overall score is selected as the word alignment for that sentence pair. Thus, for a sentence pair (e, f) we seek the alignment $\hat{a}$ such that

$$\hat{a} = \mathrm{argmax}_a \sum_{i=1}^{n} \lambda_i f_i(a, e, f)$$

where the f_i are features and the λ_i are weights.

We optimize the model weights using a modified version of averaged perceptron learning as described by Collins (2002). This is fast to train, because selecting the feature weights is the last step in building the model and the "online" nature of perceptron learning allows the parameter optimization to converge quickly. Furthermore, no generative story has to be invented to explain how the features generate the data, so new features can be easily added without having to change the overall structure of the model.

In theory, a disadvantage of a discrimintative approach compared to a generative approach is that it requires annotated data for training. In practice, however, effective discriminative models for word alignment require only a few parameters, which can be optimized on a set of annotated sentence pairs comparable in size to what is needed to tune the free parameters used in the generative approach. As we will show, a simple sequence of two such models can achieve alignment accuracy comparable to that of a combination of more complex standard models.

2 Discriminative Alignment Models

We develop two word alignment models, incorporating different word association features intended to indicate how likely two words or groups of words are to be mutual translations, plus additional features measuring how much word reordering is required by the alignment[1], and how many words are left unlinked. One of the models also includes a feature measuring how often one word is linked to several words.

Each of our feature scores have analogs in the IBM and HMM models. The association scores corresponds to word translation probabilities; the reordering scores correspond to distortion probabilities; the scores for words left unlinked corresponds to probabilities of words being linked to the null word; and the scores for one-to-many links correspond to fertility probabilities.

2.1 The Log-Likelihood-Based Model

In our first model, we use a log-likelihood-ratio (LLR) statistic as our measure of word association. We chose this statistic because it has previously been found to be effective for automatically constructing translation lexicons (e.g., Melamed, 2000). We compute LLR scores using the following formula presented by Moore (2004):

$$LLR(f, e) = \sum_{f? \in \{f, \neg f\}} \sum_{e? \in \{e, \neg e\}} C(f?, e?) \log \frac{p(f?|e?)}{p(f?)}$$

In this formula f and e mean that the words whose degree of association is being measured occur in the respective target and source sentences of an aligned sentence pair, $\neg f$ and $\neg e$ mean that the corresponding words do not occur in the respective sentences, $f?$ and $e?$ are variables ranging over these values, and $C(f?, e?)$ is the observed joint count for the values of $f?$ and $e?$. All the probabilities in the formula refer to maximum likelihood estimates. The LLR score for a pair of words is high if the words have either a strong positive association or a strong negative association. Since we expect translation pairs to be positively associated, we discard any negatively associated word pairs by requiring that $p(f, e) > p(f) \cdot p(e)$. To reduce the memory requirements of our algorithms we discard any word pairs whose LLR score is less than 1.0.

[1]We will use the term "alignment" to mean an overall word alignment of a sentence pair, and the term "link" to mean the alignment of a particular pair of words or small group of words.

In our first model, the value of the *word association feature* for an alignment is simply the sum of all the individual LLR scores for the word pairs linked by the alignment. The LLR-based model also includes the following features:

nonmonotonicity features It may be observed that in closely related languages, word alignments of sentences that are mutual translations tend to be approximately monotonic (i.e., corresponding words tend to be in nearly corresponding sentence positions). Even for distantly related languages, the number of crossing links is far less than chance, since phrases tend to be translated as contiguous chunks. To model these tendencies, we introduce two *nonmonotonicity features*.

To find the points of nonmonotonicity of a word alignment, we arbitrarily designate one of the languages as the source and the other as the target. We sort the word pairs in the alignment, first by source word position, and then by target word position. We then iterate through the sorted alignment, looking only at the target word positions. The points of nonmonotonicity in the alignment will be the places where there are backward jumps in this sequence of target word positions. For example, suppose we have the sorted alignment ((1,1)(2,4)(2,5)(3,2)(5,6)). The sequence of target word positions in this sorted alignment is (1,4,5,2,6); hence, there is one point of nonmonotonicity where target word position 2 follows target word position 5.

We still need to decide how to measure the degree of nonmonotonicity of an alignment. Two methods immediately suggest themselves. One is to sum the magnitudes of the backward jumps in the target word sequence; the other is to simply count the number of backward jumps. Rather than choose between them, we use both features.

the one-to-many feature It has often been observed that word alignment links tend to be one-to-one. Indeed, word alignment results can often be improved by restricting more general models to permit only one-to-one links. For example, Och and Ney (2003) found that the intersection of the alignments found training the IBM models in both directions always outperformed either direction alone in their experiments. Since the IBM models allow one-to-many links only in one direction, this intersection can contain only one-to-one links.

To model the tendency for links to be one-to-one, we define a *one-to-many feature* as the number of links connecting two words such that exactly one of them participates in at least one other link. We also define a *many-to-many feature* as the number of links that connect two words that both participate in other links. We don't use this directly in the model, but to cut down on the number of alignments we need to consider, we discard any alignments having a non-zero value of the many-to-many feature.

the unlinked word feature To control the number of words that get linked to something, we introduce an *unlinked word feature* that simply counts the total number of unlinked words in both sentences in an aligned sentence pair.

2.2 The Conditional-Link-Probability-Based Model

In this model we replace the LLR-based word association statistic with the logarithm of the estimated conditional probability of two words (or combinations of words) being linked, given that they co-occur in a pair of aligned sentences. These estimates are derived from the best alignments according to some other, simpler model. For example, if *former* occurs 1000 times in English sentences whose French translations contain *ancien*, and the simpler alignment model links them in 600 of those sentence pairs, we might estimate the conditional link probability (CLP) for this word pair as 0.6. We find it better, however, to adjust these probabilities by subtracting a small fixed discount from the link count:

$$LP_d(f,e) = \frac{links_1(f,e) - d}{cooc(f,e)}$$

$LP_d(f,e)$ represents the estimated conditional link probability for the words f and e, $links_1(f,e)$ is the number of times they are linked by the simpler alignment model, d is the discount, and $cooc(f,e)$ is the number of times they co-occur. This adjustment prevents assigning high probabilities to links between pairs of words that rarely co-occur.

An important difference between the LLR-based model and CLP-based model is that the LLR-based 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). In the CLP-based model, however, we allow conditional probabilities for both one-to-one and one-to-many clusters, but we require all clusters to be disjoint.

For example, we estimate the conditional probability of linking *not* to *ne...pas* by considering the number of sentence pairs in which *not* occurs in the English sentence and both *ne* and *pas* occur in the French sentence, compared to the number of times *not* is linked to both *ne* and *pas* in pairs of corresponding sentences. However, when we make this estimate in the CLP-based model, we do not count a link between *not* and *ne...pas* if the same instance of *not*, *ne*, or *pas* is linked to any other words.

The CLP-based model incorporates the same adtional features as the LLR-based model, except that it omits the one-to-many feature, since we assume that the one-to-one vs. one-to-many trade-off is already modeled in the conditional link probabilities for particular one-to-one and one-to-many clusters.

We have developed two versions of the CLP-based model, using two different estimates for the conditional link probabilities. One estimate of the conditional link probabilities comes from the LLR-based model described above, optimized on an annotated development set. The other estimate comes from a heuristic alignment model that we previously developed (Moore, 2005).[2] Space does not permit a full description of this heuristic model here, but in brief, it utilizes a series of greedy searches inspired by Melamed's competitive linking algorithm (2000), in which constraints limiting alignments to being one-to-one and monotonic are applied at different thresholds of the LLR score, with a final cutoff of the LLR score below which no alignments are made.

3 Alignment Search

While the discriminative models presented above are very simple to describe, finding the optimal alignment according to these models is non-trivial. Adding a link for a new pair of words can affect the nonmonotonicity scores, the one-to-many score, and the unlinked word score differently, depending on what other links are present in the alignment. Nevertheless, we have found a beam-search procedure that seems highly effective in finding good alignments when used with these models.

For each sentence pair, we create a list of association types and their corresponding scores, consisting of the associations for which we have determined a score and for which the words involved in the association type occur in the sentence pair.[3] We sort the resulting list of association types from best to worst according to their scores.

Next, we initialize a list of possible alignments with the empty alignment, assigning it a score equal to the number of words in the sentence pair multiplied by the unlinked word weight. We then iterate through our sorted list of association types from best to worst, creating new alignments that add links for all instances of the association type currently being considered to existing alignments, potentially keeping both the old and new alignments in our set of possible alignments.

Without pruning, we would soon be overwhelmed by a combinatorial explosion of alignments. The set of alignments is therefore pruned in two ways. First, we keep track at all times of the score of the best alignment we have seen so far, and any new alignment whose overall score is worse than the best score so far by more than a fixed difference D is immediately discarded. Second, for each instance of a particular alignment type, when we have completed creating modified versions of previous alignments to include that instance, we merge the set of new alignments that we have created into the set of previous alignments. When we do this merge, the resulting set of alignments is sorted by overall score, and only the N best alignments are kept, for a fixed N.

Some details of the search differ between the LLR-based model and the CLP-based model. One difference is how we add links to existing alignments. In both cases, if there are no existing links involving any of the words involved in the new link, we simply add it (keeping a copy of the original alignment, subject to pruning).

If there are existing links involving word instances also involved in the new link, the two mod-

[2]The conditional link probabilities used in the current work are those used in Method 4 of the earlier work. Full details are provided in the reference.

[3]By *association type* we mean a possible link between a pair of words, or, in the case of the CLP-based models, a possible one-to-many or many-to-one linkage of words.

els are treated differently. For the CLP-based model, each association score is for a cluster of words that must be disjoint from any other association cluster, so when we add links for a new cluster, we must remove any other links involving the same word instances. For the LLR-based model, we can add additional links without removing old ones, but the resulting alignment may be worse due to the degradation in the one-to-many score. We therefore add both an alignment that keeps all previous links, and an additional set of alignments, each of which omits one of the previous links involving one of the word instances involved in the new link.

The other difference in how the two models are treated is an extra pruning heuristic we use in the LLR-based model. In generating the list of association types to be used in aligning a given sentence pair, we use only association types which have the best association score for this sentence pair for one of the word types involved in the association. We initially explored limiting the number of associations considered for each word type simply as an efficiency heuristic, but we were surprised to discover that the most extreme form of such pruning actually reduced alignment error rate over any less restrictive form or not pruning on this basis at all.

4 Parameter Optimization

We optimize the feature weights using a modified version of averaged perceptron learning as described by Collins (2002). Starting with an initial set of feature weight values, perceptron learning iterates through the annotated 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 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:

$$\lambda_i \leftarrow \lambda_i + (f_i(a_{ref}, e, f) - f_i(a_{hyp}, e, f))$$

The updated feature weights are used to compute a_{hyp} for the next sentence pair.

Iterating through the data continues until the weights stop changing, because $a_{ref} = a_{hyp}$ for each sentence pair, or until some other stopping condition is met. 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.

We make a few modifications to the procedure as described by Collins. First, we average the weight values over each pass through the data, rather than over all passes, as we found this led to faster convergence. After each pass of perceptron learning through the data, we make another pass through the data with feature weights fixed to their average values for the previous learning pass, to evaluate current performance of the model. We iterate this procedure until a local optimum is found.

Next, we used a fixed weight of 1.0 for the word-association feature, which we expect to be most important feature in the model. 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. This necessitates, however, employing a version of perceptron learning that uses a learning rate parameter. As described by Collins, the perceptron update rule involves incrementing each weight by the difference in the feature values being compared. If the feature values are discrete, however, the minimum difference may be too large compared to the unweighted association score. We therefore multiply the feature value difference by a learning rate parameter η to allow smaller increments when needed:

$$\lambda_i \leftarrow \lambda_i + \eta(f_i(a_{ref}, e, f) - f_i(a_{hyp}, e, f))$$

For the CLP-based model, based on the typical feature values we expected to see, we guessed that 0.01 might be a good value for the learning rate parameter. That seemed to produce good results, so we did not attempt to further optimize the learning rate parameter for this model.

The situation with the LLR-based model was more complicated. Our previous experience using LLR scores in statistical NLP applications indicated that with large data sets, LLR values can get very high (upwards of 100000 for our 500000 sentence pair corpus), but small difference could be significant, which led us to believe that the same would be true of the weight values we were trying to learn. That meant that a learning rate small enough to let us converge on the desired weight values might take a very large number of iterations through the data to reach those values. We addressed this problem, by using a progression of learning rates, starting at 1000, reducing each successive weight by an order of magnitude, until we ended with a learning rate of 1.0. At each transition between learning rates, we re-initialized the weights to the optimum values found with the previous learning rate.

We experimented with one other idea for optimizing the weight values. Perceptron learning does not directly optimize error rate, but we have only a small number of parameters that we need to optimize. We therefore thought it might be helpful to apply a general optimization procedure directly to the error rate, starting from the best parameter values found by perceptron learning, using the N-best alignments found with these parameter values. We experimented with both the downhill simplex method (Press et al., 2002, Section 10.4) and Powell's method (Press et al., 2002, Section 10.5), but we obtained slightly better results with a more heuristic method designed to look past minor local minima. We found that using this approach on top of perceptron learning led to slightly lower error rates on the development set with the CLP-based model, but not with the LLR-base model, so we used it only with the former in our final evaluations.

5 Data and Methodology for Evaluation

We evaluated our models using data from the bilingual word alignment workshop held at HLT-NAACL 2003 (Mihalcea and Pedersen, 2003). We used a subset of the Canadian Hansards bilingual corpus supplied for the workshop, comprising 500,000 English-French sentences pairs, including 447 manually word-aligned sentence pairs designated as test data. The test data annotates particular pairs of words either as "sure" or "possible" links. Automatic sentence alignment of the training data was provided by Ulrich Germann, and the hand alignments of the words in the test data were created by Franz Och and Hermann Ney (Och and Ney, 2003).

Since our discriminative training approach requires a small amount of annotated data for parameter optimization, we split the test data set into two virtually equal subsets, by randomly ordering the test data pairs, and assigning alternate pairs from the random order to the two subsets. We used one of these subsets as a development set for parameter optimization, and held out the other for a final test set.

We report the performance of our alignment models in terms of precision, recall, and alignment error rate (AER) as defined by Och and Ney (2003):

$$\text{recall} = \frac{|A \cap S|}{|S|}$$

$$\text{precision} = \frac{|A \cap P|}{|A|}$$

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

In these definitions, S denotes the set of alignments annotated as sure, P denotes the set of alignments annotated possible or sure, and A denotes the set of alignments produced by the method under test. Following standard practice in the field, we take AER, which is derived from F-measure, as the primary evaluation metric that we are attempting to optimize.

6 Experimental Results

We first trained the LLR-based model by perceptron learning, using an N-best value of 20 and an unbounded allowable score difference in the alignment search, using the development set as annotated training data. We then aligned all the sentences of length 100 or less in our 500,000 sentence pair corpus, using an N-best value of 20 and a maximum allowable score difference of 125000. We collected link counts and co-occurrence counts from these alignments for estimating conditional link probabilities. We trained CLP-based models from these counts for a range of values for the discount used in the conditional link probability estimation, finding a value of 0.4 to be a roughly optimal value of the discount parameter for the development set. We also trained a CLP-based model using the conditional link probabilities from the heuristic alignment model mentioned previously. In training both CLP-based models, we also used an N-best value of 20 and an unbounded allowable score difference in the alignment search.

We evaluated three models on the final test data: the LLR-based model (LLR) and the two CLP-based models, one with conditional link probabilities from

Alignment	Recall	Precision	AER
LLR	0.829	0.848	0.160
CLP_1	0.889	0.934	0.086
CLP_2	0.898	0.947	0.075

Table 1: Discriminative Model Results.

Alignment	Recall	Precision	AER
$\text{E} \rightarrow \text{F}$	0.870	0.890	0.118
$\text{F} \rightarrow \text{E}$	0.876	0.907	0.106
Union	0.929	0.845	0.124
Intersection	0.817	0.981	0.097
Refined	0.908	0.929	0.079

Table 2: IBM Model 4 Results.

the LLR-based model (CLP_1), and one with conditional link probabilities from the heuristic alignment model (CLP_2). All parameters were optimized on the development set. Recall, precision, and alignment error rates on the test set are shown in Table 1.

For comparison, we aligned our parallel corpus with IBM Model 4 using Och's Giza++ software package (Och and Ney, 2003).[4] We used the default configuration file included with the version of Giza++ that we used, which resulted in five iterations of Model 1, followed by five iterations of the HMM model, followed by five iterations of Model 4. We trained the models in both directions, English-to-French and French-to-English, and computed the union, intersection, and what Och and Ney (2003) call the "refined" combination of the two alignments. We evaluated the resulting alignments of the final test set, with the results shown in Table 2.

As these tables show, our discriminatively trained CLP-based models compare favorably to IBM Model 4 on this data set. The one with conditional link probabilities from the heuristic alignment model, CLP_2, performs slightly better than the best of the Model 4 combinations, and the one with conditional link probabilities from the LLR-based model, CLP_1, performs only slightly worse.

An interesting question is why CLP_2 outperformed CLP_1. CLP_1 is the more "principled" model, so one might have expected it to perform better. We believe the most likely explanation is the fact that CLP_2 received 403,195 link probabilities from the heuristic model, while CLP_1 received only 144,051 link probabilities from the LLR-based model. Hence CLP_2 was able to consider more possible links.

In light of our claims about the ease of optimizing the models, we should make some comments on the time need to train the parameters. Our current implementation of the alignment search is written in Perl, and is therefore quite slow. Alignment of our 500,000 sentence pair corpus with the LLR-based mode took over a day on a 2.8 GHz Pentium IV workstation. Nevertheless, the parameter optimization was still quite fast, since it took only a few iterations over our 224 sentence pair development set. With either the LLR-based or CLP-based models, one combined learning/evaluation pass of perceptron training always took less than two minutes, and it never took more that six passes to reach the local optimum we took to indicate convergence. Total training time was greater since we used multiple runs of perceptron learning with different learning rates for the LLR-based model and different conditional link probability discounts for CLP_1, but total training time for each model was around an hour.

7 Related Work

When the first version of this paper was submitted for review, we could honestly state, "We are not aware of any previous work on discriminative word alignment models." Callison-Burch et al. (2004) had investigated the use of small amounts of annotated data to help train the IBM and HMM models, but the models were still generative and were trained using maximum-likelihood methods.

Recently, however, three efforts nearly simultaneous with ours have made use of discriminative methods to train alignment models. Fraser and Marcu (2005) modify Model 4 to be a log-linear combination of 11 submodels (5 based on standard Model 4 parameters, and 6 based on additional features) and discriminatively optimize the submodel weights on each iteration of a Viterbi approximation to EM.

Liu et al. (2005) also develop a log-linear model, based on IBM Model 3. They train Model 3 using Giza++, and then use the Model 3 score of a possible alignment as a feature value in a discriminatively trained log-linear model, along with fea-

[4]Thanks to Chris Quirk for carrying out this alignment.

tures incorporating part-of-speech information, and whether the aligned words are given as translations in a bilingual dictionary. The log-linear model is trained by standard maximum-entropy methods.

Klein and Taskar (2005), in a tutorial on maximum margin methods for natural-language processing, described a weighted linear model incorporating association, position, and orthography features, with its parameters trained by a structured-support-vector-machine method. This model is in some respects very similar to our LLR-based model, using Dice coefficient association scores where we use LLR scores, and absolute position differences where we use nonmonotonicity measures.

8 Conclusions

The results of our work and other recent efforts on discriminatively trained alignment models show that results comparable to or better than those obtained with the IBM models are possible within a framework that makes it easy to add arbitrary additional features. After many years using the same small set of alignment models, we now have an easy way to experiment with a wide variety of knowledge sources to improve word-alignment accuracy.

References

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.

Chris Callison-Burch, David Talbot, and Miles Osborne. 2005. Statistical Marchine Translation with Word- and Sentences-Aligned Parallel Corpora. In *Proceedings of the 42nd Annual Meeting of the ACL*, pp. 176–183, Barcelona, Spain.

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.

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.

Dan Klein and Ben Taskar. 2005. Max-Margin Methods for NLP: Estimation, Structure, and Applications. Tutorial presented at ACL 2005, Ann Arbor, Michigan.

I. Dan Melamed. 2000. Models of Translational Equivalence. *Computational Linguistics*, 26(2):221–249.

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, Canada.

Robert C. Moore. 2004. On Log-Likelihood-Ratios and the Significance of Rare Events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 333–340, Barcelona, Spain.

Robert C. Moore. 2005. Association-Based Bilingual Word Alignment. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pp. 1–8, Ann Arbor, Michigan.

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):417–449.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipies in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, Cambridge, England.

A Maximum Entropy Word Aligner for Arabic-English Machine Translation

Abraham Ittycheriah and Salim Roukos
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
{abei,roukos}@us.ibm.com

Abstract

This paper presents a maximum entropy word alignment algorithm for Arabic-English based on *supervised* training data. We demonstrate that it is feasible to create training material for problems in machine translation and that a mixture of supervised and unsupervised methods yields superior performance. The probabilistic model used in the alignment directly models the link decisions. Significant improvement over traditional word alignment techniques is shown as well as improvement on several machine translation tests. Performance of the algorithm is contrasted with human annotation performance.

1 Introduction

Machine translation takes a source sequence,

$$S = [s_1 \ s_2 \ \dots \ s_K]$$

and generates a target sequence,

$$T = [t_1 \ t_2 \ \dots \ t_M]$$

that renders the meaning of the source sequence into the target sequence. Typically, algorithms operate on sentences. In the most general setup, one or more source words can generate 0, 1 or more target words. Current state of the art machine translation systems (Och, 2003) use phrasal (n-gram) features extracted automatically from parallel corpora. These phrases are extracted using word alignment algorithms that are trained on parallel corpora. Phrases, or phrasal features, represent a mapping of source sequences into a target sequences which are typically a few words long.

In this paper, we investigate the feasibility of training alignment algorithms based on supervised alignment data. Although there is a modest cost associated with annotating data, we show that a reduction of 40% relative in alignment error (AER) is possible over the GIZA++ aligner (Och and Ney, 2003).

Although there are a number of other applications for word alignment, for example in creating bilingual dictionaries, the primary application continues to be as a component in a machine translation system. We test our aligner on several machine translation tests and show encouraging improvements.

2 Related Work

Most of the prior work on word alignments has been done on parallel corpora where the alignment at the sentence level is also done automatically. The IBM models 1-5 (Brown et al., 1993) produce word alignments with increasing algorithmic complexity and performance. These IBM models and more recent refinements (Moore, 2004) as well as algorithms that bootstrap from these models like the HMM algorithm described in (Vogel et al., 1996) are unsupervised algorithms.

The relative success of these automatic techniques together with the human annotation cost has delayed the collection of supervised word-aligned corpora for more than a decade.

(Cherry and Lin, 2003) recently proposed a direct alignment formulation and state that it would be straightforward to estimate the parameters given a supervised alignment corpus. In this paper, we extend their work and show that with a small amount of annotated data, together with a modeling strategy and search algorithm yield significant gains in alignment F-measure.

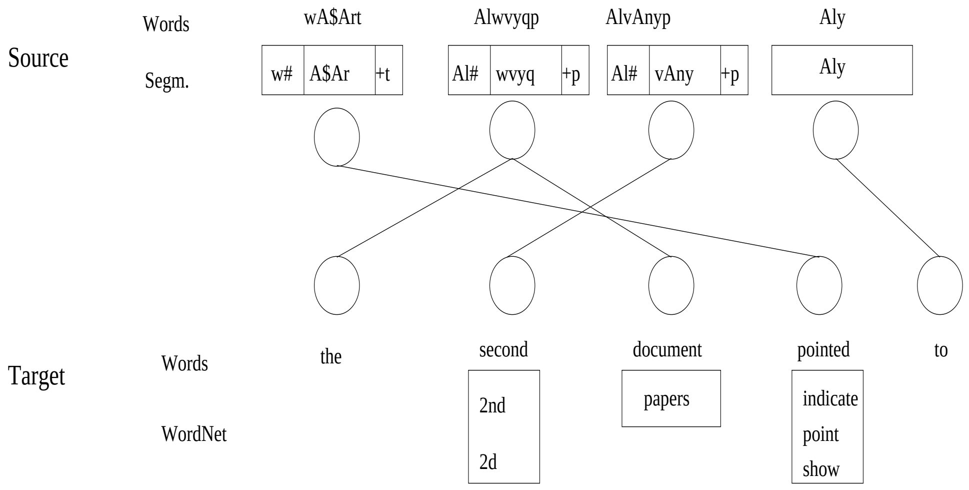


Figure 1: Alignment example.

3 Algorithm

In order to describe the algorithm, we will need to first describe the direct link model. Figure 1 shows two sequences where the top sequence is considered the source sequence and the bottom sequence the target sequence. Each sequence can have auxilliary information such as Arabic segmentation or English WordNet (Miller, 1990) information as shown. Given the source and target sequences, there are a number of different ways to link each target word to a source word. Each target word has a link l_i which indicates which source position it links to. The range of l_i is from 0 to K and there are M of these links. The source word position 0 is used to indicate NULL which we imagine gives rise to unaligned English words. In this paper, we refer to these words as being spontaneous. A valid link configuration has M links. Define $\mathcal{L}$ to be the set of all possible valid link configurations, and L to be a member of that set. We seek to maximize the alignment probability by finding the optimum link configuration L_{opt},

$$\begin{aligned} p(L_{\mathrm{opt}}|S,T) &= \arg\max_{L\in\mathcal{L}} p(L|S,T) \\ &= p(l_i^M|t_1^M, s_1^K) \\ &= \prod_{i=0}^{M} p(l_i|t_1^M, s_1^K, l_1^{i-1}). \end{aligned}$$

We factor this into a transition model and an observation model,

$$p(L|S,T) = \frac{1}{Z}\prod_{i=0}^{M} p(l_i|l_{i-1})^{\alpha} p(l_i|t_1^M, s_1^K, l_1^{i-1})^{1-\alpha}.$$

where Z is the normalizing constant.

We factor the model as above so that the transition model computation, which uses information available on the search hypotheses, is reduced during the search process. In the aligner presented here, α is always set to 0.5. Next we will describe the transition model, then the observation model and finally the experiments in alignment and machine translation.

In the IBM Model 1 aligner, the choice of the language to serve as states of the search algorithm is not prescribed, but practically the choice is important as it affects performance. To see this, note that in generative models an input word can only be aligned to a single state in the search. In our current situation, we are interested in aligning unsegmented Arabic words and typical words have a few affixes to indicate for example pronouns, definiteness, prepositions and conjunctions. In English these are separate words, and therefore to maximize performance the unsegmented Arabic words serve as states in the search algorithm and we align English words to these states.

3.1 Transition Model

The transition model tends to keep the alignments close together and penalizes alignments in which adjacent words in the target language come from very distant words in the source language. Also, we would like to penalize many English words coming from the same Arabic state; we call this the state visit penalty and will be described later. In this paper, we use a parametric form for the transition model,

$$p(l_i|l_{i-1}) = \frac{1}{Z(l_{i-1})}\left[\frac{1}{\mathrm{dist}(l_i, l_{i-1})} + \frac{1}{ns(l_i)}\right] \quad (1)$$

where $ns(i)$ represents the state visit penalty for state i, $Z(l_{i-1})$ is the normalization constant and

$$\mathrm{dist}(l_i, l_{i-1}) = \min(|l_i - l_{i-1}|, |l_i - f_i|) + a.$$

Here a is a penalty for a zero distance transition and is set to 1 in the experiments below. The min operator chooses the lowest cost transition distance either from the previous state or the frontier state, f_i, which is the right most state that has been visited (even though Arabic is normally displayed right to left, we make our Arabic state graphs from left to right). This is a language specific criteria and intended to model the adjective noun reversal between English and Arabic. Once the current noun phrase is completed, the next word often aligns to the state just beyond frontier state. As an example, in Figure 1, the verb 'pointed' aligns to the first Arabic word 'wA$Art', and aligning the 'to' to its Arabic counterpart 'Aly' would incur normally a distance of 3 but with the frontier notion it incurs only a penalty of 1 on the hypothesis that aligns the word 'second' to 'AlvAnyp'. In this alignment with the frontier notion, there are only distance 1 transitions, whereas the traditional shapes would incur a penalty of 2 for alignment of 'pointed' and a penalty of 3 for the word 'to'.

The state visit penalty, $ns(i)$ is the distance between the English words aligned to this state times the number of state visits[1]. This penalty controls the fertility of the Arabic words. To determine the English words that aligned to the Arabic position, the search path is traced back for each hypothesis and a sufficiently large beam is maintained so that alignments in the future can correct past alignment decisions. This penalty allows English determiners and prepositions to align to the Arabic content word while penalizing distant words from aligning to the state. In terms of alignment F-measure to be described below, the state visit penalty, if removed makes the performance degrade from F=87.8 to F=84.0 compared to removing the frontier notion which only degrades performance to F=86.9.

3.2 Observation Model

The observation model measures the linkage of the source and target using a set of feature functions defined on the words and their context. In Figure 1, an event is a single link from an English word to an Arabic state and the event space is the sentence pair. We use the maximum entropy formulation (e.g. (Berger et al., 1996)),

$$f = \psi(l_i)$$
$$h = \left[t_1^{i-1}, s_1^K\right]$$
$$p(f|h) = \frac{1}{Z(h)} \exp \sum_i \lambda_i \phi_i(h, f),$$

where Z(h) is the normalizing constant,

$$Z(h) = \sum_f \exp \sum_i \lambda_i \phi_i(h, f).$$

and $\phi_i(h, f)$ are binary valued feature functions. The function ψ selects the Arabic word at the position being linked or in the case of segmentation features, one of the segmentations of that position. We restrict the history context to select from the current English word and words to the left as well as the current word's WordNet (Miller, 1990) synset as required by the features defined below. As in (Cherry and Lin, 2003), the above functions simplify the conditioning portion, h by utilizing only the words and context involved in the link l_i. Training is done using the IIS technique (Della Pietra et al., 1995) and convergence often occurs in 3-10 iterations. The five types of features which are utilized in the system are described below.

Phrase to phrase (for example, idiomatic phrases) alignments are intepreted as each English word coming from each of the Arabic words.

3.2.1 Lexical Features

The lexical features are similar to the translation matrix of the IBM Model 1. However, there is a signficant out of vocabulary (OOV) issue in the model since training data is limited. All words that have a corpus frequency of 1 are left out of the model and classed into an unknown word class in order to explicitly model connecting unknown words. From the training data we obtain 50K lexical features, and applying the Arabic segmenter obtain another 17K lexical features of the form ϕ(English content word, Arabic stem).

3.2.2 Arabic Segmentation Features

An Arabic segmenter similar to (Lee et al., 2003) provides the segmentation features. A small dictionary is used (with 71 rules) to restrict the set of Arabic segments that can align to English stopwords, for example that 'the' aligns to 'Al#' and that 'for', 'in' and 'to' align to 'b#' and 'her' aligns with the suffix '+hA'. Segmentation features also help align unknown words, as stems might be seen in the training corpus with other prefixes or suffixes. Additionally, the ability to align the prefix and suffix accurately, tends to 'drag' the unknown stem to its English target.

[1]We are overloading the word 'state' to mean Arabic word position.

3.2.3 WordNet Features

WordNet features provide normalization on the English words. The feature is instantiated for nouns, adjectives, adverbs and verbs following their definitions in WordNet. If the Arabic word has a segmentation then the feature is ϕ(WordNet synset id, Arabic stem), otherwise it is ϕ(WordNet synset id, Arabic word). The feature ties together English synonyms and helps improve recall of the aligner.

3.2.4 Spelling Feature

The spelling feature is applied only on unknown words and is used to measure the string kernel distance(Lodhi et al., 2000) between romanized Arabic and English words. The feature is designed primarily to link unknown names. For example, 'Clinton' is written as 'klyntwn' in one of its romanized Arabic versions. In a sentence, measuring the string kernel distance shows a correlation between these names even though there is not much overlap between the characters. The feature has four possible values: nomatch, somematch, goodmatch, and exact.

3.2.5 Dynamic Features

Dynamic features are defined on the lattice of the search algorithm. These features fire when the previous source and target word pair are linked. For example, one such feature is 'b# in' and if on the hypothesis we have just linked this pair and the next English word is being aligned to the stem of the Arabic word where this prefix occurs, this feature fires and boosts the probability that the next words are aligned. The basic intuition behind this feature is that words inside prepositional phrases tend to align, which is similar to the dependency structure feature of (Cherry and Lin, 2003).

At training time, the lattice reduces to the single path provided by the annotation. Since this feature tends to suffer from the drag of function words, we insist that the next words that are being linked have at least one feature that applies. All word pairs linked in the training data have lexical features as described above, and if both source and target words are unknown they have a single feature for their link. Applying dynamic features on words that have at least one other feature prevents words which are completely unrelated from being linked because of a feature about the context of the words.

Two types of dynamic features are distinguished: (a) English word with Arabic prefix/suffix and (b) English word with Arabic stem.

4 Smoothing the Observation Model

Since the annotated training data for word alignment is limited and a much larger parallel corpus is available for other aligners, we smooth the observation probability with an IBM Model 1 estimate,

$$p(l_i|t_1^M, s_1^K) = \frac{1}{Z} p_{\mathrm{ME}}(l_i|t_1^M, s_1^K)^{\beta} p_{\mathrm{M1}}(s|t_i)^{1-\beta}.$$

where β is set to 0.9 in the experiments below. In the equation above, the s represents the Arabic word that is being linked from the English word t_i.

When β is set to 1.0 there is no smoothing performed and performance degrades to F=84.0 from the best system performance (F=87.8). When β is set to 0, the model uses only the IBM Model 1 distribution and the resulting aligner is similar to an HMM aligner with the transition shape discussed above and yields performance of F=73.2.

	Anno. 1 Correction	Anno. 1'	Anno. 2
Anno. 1	96.5	92.4	91.7
Anno. 1'	95.2	—	93.2

Table 1: F-measure for human performance on word alignment for Arabic-English.

5 Search Algorithm

A beam search algorithm is utilized with the English words consumed in sequence and the Arabic word positions serving as states in the search process. In order to take advantage of the transition model described above, a large beam must be maintained. To see this, note that English words often repeat in a sentence and the models will tend to link the word to all Arabic positions which have the same Arabic content. In traditional algorithms, the Markov assumption is made and hypothesis are merged if they have the same history in the previous time step. However, here we maintain all hypotheses and merge only if the paths are same for 30 words which is the average sentence length.

6 Experimental Data

We have word aligned a portion of the Arabic Treebank (4300 sentences) and material from the LDC news sources (LDC, 2005) to obtain a total of 10.3K sentence pairs for training. As a test of alignment, we use the first 50 sentences of the MT03 Evaluation test set which has 1313 Arabic words and 1528 English words [2]. In terms of annotation guidelines, we use the following instructions: (a) Align determiners to their head nouns, (b) Alignments are done word by word unless the phrase is idiomatic in which case the entire phrase to phrase alignment was marked, (c) spontaneous words are marked as being part of a

[2]The test data is available by contacting the authors.

	1K	3K	5K	7K	9K	10.3K
# of features	15510	32111	47962	63140	73650	80321
English % OOV	15.9	8.2	5.5	4.4	4.05	3.6
Arabic % OOV	31	19.6	15.6	13.2	10.8	10.3
F-measure	83.2	85.4	86.5	87.4	87.5	87.8

Table 2: Varying Training data size.

phrase wherever possible but left unaligned if there is no evidence to link the word.

In order to measure alignment performance, we use the standard AER measure (Och and Ney, 2000) but consider all links as sure. This measure is then related to the F-measure which can be defined in terms of precision and recall as

Precision The number of correct word links over the total number of proposed links.

Recall The number of correct word links over the total number of links in the reference.

and the usual definition of the F-measure,

$$F = \frac{2PR}{(R + P)}$$

and define the alignment error as AER $= 1 - F$. In this paper, we report our results in terms of F-measure over aligned links. Note that links to the NULL state (unaligned English words) are not included in the F-measure. Systems are compared relative to the reduction in AER.

6.1 Annotator Agreement

We measure intra/inter-annotator agreement on the test set in order to determine the feasibility of human annotation of word links. These are shown in Table 1. In the table, the column for 'Annotator 1 Correction' is the first annotator correcting his own word alignments after a span of a year. After two weeks, the annotator (Annotator 1') was given the same material with all the links removed and asked to realign and we see that there is more discrepancy in resulting alignments. The differences are largely on the head concept where determiners are attached and the alignment of spontaneous words. The performance with a second annotator is in the same range as the reannotation by a single annotator.

7 Experiments

In order to evaluate the performance of the algorithm, we investigate the effect due to: (a) increasing the training data size, (b) additional feature types, and (c) comparable algorithms.

7.1 Training Data Size

We varied the training data size from 1K sentences to the complete set in Table 2. Each batch re-estimates the unknown word class by creating a vocabulary on the training set. The trend indicates a reasonable progression of performance and more data is required to determine the saturation point.

7.2 Feature Types

The results obtained by different feature sets are shown in Table 3. Each feature type was added incrementally (Add Feature column) to the line above to determine the effect of the individual feature types and then removed incrementally from the full system (Subtract Feature column) in order to see the final effect. The results indicate that lexical features are the most important type of feature; segmentation features further reduce the AER by 15.8%. The other features add small gains in performance which, although are not statistically significant for the alignment F-measure, are important in terms of feature extraction. Segmentation features discussed above result in both suffix and prefix features as well as stem features. In the Subtract column, for the segmentation feature, only the suffix and prefix features were removed. This result indicates that most of the alignment improvement from the segmentation feature comes in the form of new lexical features to link Arabic stems and English words.

7.3 Comparison to other alignment algorithms

In order to gauge the performance of the algorithm with respect to other alignment strategies, we provide results using GIZA++ and an HMM Max Posterior Algorithm (Ge, 2004). These algorithms, as well as the Model 1 smoothing for the MaxEnt aligner, are all trained on a corpus of 500K sentence pairs from the UN parallel corpus and the LDC news corpora released for 2005 (LDC, 2005). Note that these algorithms are unsupervised by design but we utilize them to have a baseline for comparing the performance of this supervised approach.

7.3.1 HMM Max Posterior Aligner

The maximum-posterior word alignments are obtained by finding the link configuration that maxi-

System	# of feats	Add Feature	Subtract Feature
Word pairs	50070	85.03	76.3
Spelling	4	85.11	87.7
Segmentation	70	87.39	87.5(*)
WordNet	13789	87.54	87.5
Dynamic-Words	1952	87.80	87.1
Dynamic-Segmentation	42	87.84	87.8

Table 3: Alignment performance in terms of the feature types utilized.

	F-Measure
GIZA++	79.5
HMM	76.3
MaxEnt	87.8

Table 4: Alignment performance

mizes the posterior state probability. In contrast, in performing a Viterbi alignment, we compute the best state sequence given the observation. The maximum posterior computes the best state one at a time and iterates over all possible combinations. Once we find the maximum in the posterior probability matrix, we also know the corresponding state and observation which is nothing but the word pair (s_j, t_i). We will then align the pair and continue to find the next posterior maximum and align the resulting pair. At each iteration of the process, a word pair is aligned. The process is repeated until either every word in one (or both) language is aligned or no more maximum can be found, whichever happens first.

7.3.2 GIZA Alignment

In order to contrast our algorithm, we ran GIZA++ in the standard configuration which implies 5 iterations of IBM Model 1, HMM, Model 3 and Model 4. All parameters are left to their default values.

The results using the three different aligners is shown in Table 4. The reduction in AER over the GIZA++ system is 40.5% and over the HMM system is 48.5%. The Wilcoxon signed-rank test yields a probability of 0.39 for rejecting the GIZA++ alignment over the HMM alignment, whereas the MaxEnt algorithm should be rejected with a probability of 1.7e-6 over the HMM algorithm and similarly MaxEnt should be rejected with a probability of 0.9e-6 over the GIZA++ algorithm. These significance tests indicate that the MaxEnt algorithm presented above is significantly better than either GIZA++ or HMM.

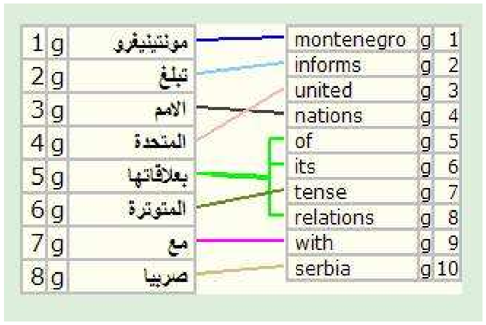


Figure 2: An alignment showing a split link from an Arabic word.

8 Phrase Extraction

Once an alignment is obtained, phrases which satisfy the inverse projection constraint are extracted (although earlier this constraint was called consistent alignments (Och et al., 1999)). This constraint enforces that a sequence of source words align to a sequence of target words as defined by the lowest and highest target index, and when the target words are projected back to the source language through the alignment, the original source sequence is retrieved. Examination of the hand alignment training data showed that this criteria is often violated for Arabic and English. Prepositional phrases with adjectives often require a split– for example, the alignment shown in Figure 2 has 'of its relations' aligned to a word in Arabic and 'tense' aligned to the next word. The inverse projection constraint fails in this case, and in the experiments below, we relax this constraint and generate features for single source words as long as the target phrase has a gap less than 2 English words. This relaxation allows a pair of adjectives to modify the head noun. In future work we explore the use of features with variables to be filled at decode time.

9 Translation Experiments

The experiments in machine translation are carried out on a phrase based decoder similar to the one de-

	MT03	MT04	MT05
GIZA++	0.454	—	—
HMM	0.459	0.419	0.456
MaxEnt	0.468	0.433	0.451
Combined	0.479	0.437	0.465
Significance	0.017	0.020	—

Table 5: Machine Translation Performance using the NIST 2005 Bleu scorer

scribed in (Tillmann and Ney, 2003). In order to contrast the performance of the extracted features, we compare the translation performance to (a) a system built from alignments proposed by an HMM Max Posterior Aligner, and (b) a system built from GIZA alignments. All other parameters of the decoder remain constant and only the feature set is changed for these experiments. As training data, we use the UN parallel corpus and the LDC news corpora released in 2005. Comparison should therefore be only made across systems reported here and not to earlier evaluations or other systems. The results are shown in Table 5.

Combination of the phrasal features from the HMM and MaxEnt alignments results in the 'Combined' system. The Combined system performs better in all cases; in MT03 and MT04 the MaxEnt derived features perform better than the HMM system. In MT05, there is a slight degradation which is not significant and the combination system still results in an improvement over either system. Since the MaxEnt aligner has access to a unique resource, every attempt was made to make that resource available to the other systems. Although GIZA++ and HMM can not directly utilize word aligned data, the training data for MaxEnt was converted to parallel sentences where each sentence has only the pair of linked words. The resulting numbers make both HMM and GIZA much closer in performance to the MaxEnt aligner but the results are better for comparing alignment methods.

10 Error Analysis and Discussion

The alignment errors made by the system can be attributed to

- English words that require multi-word Arabic states, for example (a) dates which are written in Arabic in more than one form 'kAnwn AlvAny / ynAyr' for 'january', and (b) compound words like 'rAm Allh' in English is 'Ramallah'.
- Rare translation of a common Arabic word as well as a common English word used as the translation for a rare Arabic word.
- Parallel corpora mismatch: training material for translation is processed at a document level and yet systems often operate at a sentence level. Human translators often use pronouns for earlier mentioned names although in the source language the name is repeated. Information which is sometimes repeated in the source in an earlier sentence is dropped in future sentences of the document. Document level features are required to allow the system to have information to leave these words unaligned.

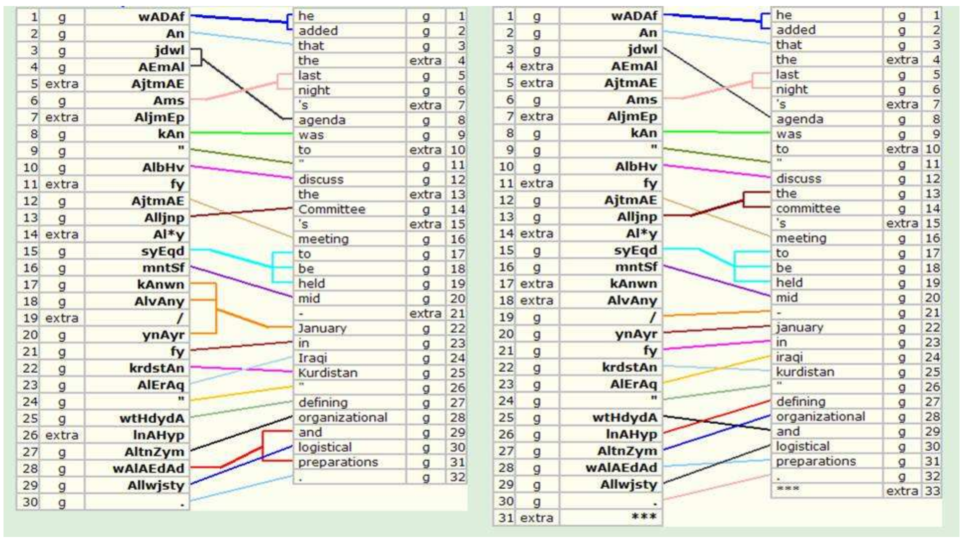
Figure 3 shows a human alignment on the left and a machine output on the right. The columns next to the words indicate whether the alignments are 'good' or 'extra' which indicates that these words are aligned to the special NULL state. There are two examples of multi-word Arabic states shown: (a) for 'january', and (b) the English word 'agenda'. The system aligns 'the' before committee and it seems in this case its an annotation error. In this example the Arabic words lnAHyp, AltnZym, wAlAEdAd and Allwjsty are all unknown words in the vocabulary yet the system managed to link 3 out 4 words correctly.

While significant gains have been made in alignment performance, these gains have not directly translated to machine translation improvements. In fact, although the GIZA system is better than the HMM system at alignment, the machine translation result on MT03 indicates a slight degradation (although it is not statistically significant). The prime reason for this is that features extracted from the alignments are aggregated over the training corpus and this process helps good alignments to have significantly better counts than errors in alignment. Aligning rare words correctly should help performance but since their count is low it is not reflected in bleu scores.

11 Conclusion and Future Work

This paper presented a word aligner trained on annotated data. While the performance of the aligner is shown to be significantly better than other unsupervised algorithms, the utility of these alignments in machine translation is still an open subject although gains are shown in two of the test sets. Since features are extracted from a parallel corpus, most of the information relating to the specific sentence alignment is lost in the aggregation of features across sentences. Improvements in capturing sentence context could allow the machine translation system to use a rare but correct link appropriately.

Another significant result is that a small amount (5K sentences) of word-aligned data is sufficient for this algorithm since a provision is made to handle



1	g	wADAf		he	g	1
2	g	An		added	g	2
3	g	jdwl		that	g	3
4	g	AEmAl		the	extra	4
5	extra	AjtmAE		last	g	5
6	g	Ams		night	g	6
7	extra	AljmEp		's	extra	7
8	g	kAn		agenda	g	8
9	g	"		was	g	9
10	g	AlbHv		to	extra	10
11	extra	fy		"	g	11
12	g	AjtmAE		discuss	g	12
13	g	Alljnp		the	extra	13
14	extra	Al*y		Committee	g	14
15	g	syEqd		's	extra	15
16	g	mntSf		meeting	g	16
17	g	kAnwn		to	g	17
18	g	AlvAny		be	g	18
19	extra	/		held	g	19
20	g	ynAyr		mid	g	20
21	g	fy		-	extra	21
22	g	krdstAn		January	g	22
23	g	AlErAq		in	g	23
24	g	"		Iraqi	g	24
25	g	wtHdydA		Kurdistan	g	25
26	extra	lnAHyp		"	g	26
27	g	AltnZym		defining	g	27
28	g	wAlAEdAd		organizational	g	28
29	g	Allwjsty		and	g	29
30	g	.		logistical	g	30
				preparations	g	31
				.	g	32

1	g	wADAf		he	g	1
2	g	An		added	g	2
3	g	jdwl		that	g	3
4	extra	AEmAl		the	extra	4
5	extra	AjtmAE		last	g	5
6	g	Ams		night	g	6
7	extra	AljmEp		's	extra	7
8	g	kAn		agenda	g	8
9	g	"		was	g	9
10	g	AlbHv		to	extra	10
11	extra	fy		"	g	11
12	g	AjtmAE		discuss	g	12
13	g	Alljnp		the	g	13
14	extra	Al*y		committee	g	14
15	g	syEqd		's	extra	15
16	g	mntSf		meeting	g	16
17	extra	kAnwn		to	g	17
18	extra	AlvAny		be	g	18
19	g	/		held	g	19
20	g	ynAyr		mid	g	20
21	g	fy		-	g	21
22	g	krdstAn		january	g	22
23	g	AlErAq		in	g	23
24	g	"		iraqi	g	24
25	g	wtHdydA		kurdistan	g	25
26	g	lnAHyp		"	g	26
27	g	AltnZym		defining	g	27
28	g	wAlAEdAd		organizational	g	28
29	g	Allwjsty		and	g	29
30	g	.		logistical	g	30
31	extra	***		preparations	g	31
				.	g	32
				***	extra	33

Figure 3: An example sentence with human output on the left and system output on the right.

unknown words appropriately.

12 Acknowledgements

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position or policy of the U.S. government and no official endorsement should be inferred. This paper owes much to the collaboration of the Statistical MT group at IBM.

References

Adam L. Berger, Vincent Della Pietra, and Stephen Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

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.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 88–95, Sapporo, Japan.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1995. Inducing features of random fields. *Technical Report, Department of Computer Science, Carnegie-Mellon University, CMU-CS-95-144*, May.

Niyu Ge. 2004. Improvement in Word Alignments. *Presentation given at DARPA/TIDES MT workshop.*

LDC. 2005. http://ldc.upenn.edu/projects/tides/mt2005ar.htm.

Young-Suk Lee, Kishore Papineni, and Salim Roukos. 2003. Language model based arabic word segmentation. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 399–406, Sapporo, Japan.

Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2000. Text classification using string kernels. In *NIPS*, pages 563–569.

G. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Robert C. Moore. 2004. Improving IBM Word-Alignment Model 1. In *42nd Annual Meeting of the Association for Computational Linguistics*, pages 518–525, Barcelona, Spain.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong, China.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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. 2003. Minimum error rate training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for Statistical Machine Translation. 29(1):97–133.

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.

A Large-Scale Exploration of Effective Global Features for a Joint Entity Detection and Tracking Model

Hal Daumé III and **Daniel Marcu**
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{hdaume,marcu}@isi.edu

Abstract

Entity detection and tracking (EDT) is the task of identifying textual mentions of real-world entities in documents, extending the named entity detection and coreference resolution task by considering mentions other than names (pronouns, definite descriptions, etc.). Like NE tagging and coreference resolution, most solutions to the EDT task separate out the mention detection aspect from the coreference aspect. By doing so, these solutions are limited to using only local features for learning. In contrast, by modeling both aspects of the EDT task simultaneously, we are able to learn using highly complex, non-local features. We develop a new joint EDT model and explore the utility of many features, demonstrating their effectiveness on this task.

1 Introduction

In many natural language applications, such as automatic document summarization, machine translation, question answering and information retrieval, it is advantageous to pre-process text documents to identify references to entities. An entity, loosely defined, is a person, location, organization or geo-political entity (GPE) that exists in the real world. Being able to identify references to real-world entities of these types is an important and difficult natural language processing problem. It involves finding text spans that correspond to an entity, identifying what *type of entity* it is (person, location, etc.), identifying what *type of mention* it is (name, nominal, pronoun, etc.) and finally identifying which *other* mentions in the document it corefers with. The difficulty lies in the fact that there are often many ambiguous ways to refer to the same entity. For example, consider the two sentences below:

$\underline{\text{Bill Clinton}}^{\text{NAM}}_{\text{PER}-1}$ gave a speech today to $\underline{\text{the Senate}}^{\text{NAM}}_{\text{ORG}-2}$. $\underline{\text{The President}}^{\text{NOM}}_{\text{PER}-1}$ outlined $\underline{\text{his}}^{\text{PRO}}_{\text{PER}-1}$ plan for budget reform to $\underline{\text{them}}^{\text{PRO}}_{\text{ORG}-2}$.

There are five entity *mentions* in these two sentences, each of which is underlined (the corresponding mention type and entity type appear as superscripts and subscripts, respectively, with coreference chains marked in the subscripts), but only two *entities*: { Bill Clinton, The president, his } and { the Senate, them }. The *mention detection* task is to identify the entity mentions and their types, without regard for the underlying entity sets, while *coreference resolution* groups a given mentions into sets.

Current state-of-the-art solutions to this problem split it into two parts: mention detection and coreference (Soon et al., 2001; Ng and Cardie, 2002; Florian et al., 2004). First, a model is run that attempts to identify each mention in a text and assign it a type (person, organization, etc.). Then, one holds these mentions fixed and attempts to identify which ones refer to the same entity. This is typically accomplished through some form of clustering, with clustering weights often tuned through some local learning procedure. This pipelining scheme has the significant drawback that the mention detection module cannot take advantage of information from the coreference module. Moreover, within the coreference task, performing learning and clustering as separate tasks makes learning rather ad-hoc.

In this paper, we build a model that solves the mention detection and coreference problems in a simultaneous, joint manner. By doing so, we are able to obtain an empirically superior system as well as integrate a large collection of features that one cannot consider in the standard pipelined approach. Our ability to perform this modeling is based on the *Learning as Search Optimization* framework, which we review in Section 2. In Section 3, we describe our joint EDT model in terms of the search procedure executed. In Section 4, we describe the features we employ in this model; these include the standard lexical, semantic (WordNet) and string matching features found in most other systems. We additionally consider many other feature types, most interestingly *count-based features*, which take into account the distribution of entities and mentions (and are not expressible in the binary classification method for coreference) and *knowledge-based features*, which exploit large corpora for learning name-to-nominal references. In Section 5, we present our experimental results. First, we compare our joint system with a pipelined version of the system, and show that joint inference leads to improved performance. Next, we perform an extensive feature comparison experiment to determine which features are most useful for the coreference task, showing that our newly introduced features provide useful new information. We conclude in Section 6.

2 Learning as Search Optimization

When one attempts to apply current, standard machine learning algorithms to problems with combinatorial structured outputs, the resulting algorithm implicitly assumes that it is possible to find the best structures for a given input (and some model parameters). Furthermore, most models require much more, either in the form of feature expectations for conditional likelihood-based methods (Lafferty et al., 2001) or local marginal distributions for margin-based methods (Taskar et al., 2003). In many cases—including EDT and coreference—this is a false assumption. Often, we are not able to find the *best* solution, but rather must employ an approximate search to find the best possible solution, given time and space constraints. The Learning as Search Optimization (LaSO) framework exploits this difficulty as an opportunity and seeks to find model parameters that are good *within the context of search.*

```
Algo Learn(problem, initial, enqueue, $\boldsymbol{w}$, $x$, $y$)
nodes ← MakeQueue(MakeNode(problem,initial))
while nodes is not empty do
  node ← RemoveFront(nodes)
  if none of nodes ∪ {node} is $y$-good or
      GoalTest(node) and node is not $y$-good then
      sibs ← siblings(node, $y$)
      $\boldsymbol{w}$ ← update($\boldsymbol{w}$, $x$, sibs, node ∪ nodes)
      nodes ← MakeQueue(sibs)
  else
      if GoalTest(node) then return $\boldsymbol{w}$
      next ← Operators(node)
      nodes ← enqueue(problem, nodes, next, $\boldsymbol{w}$)
  end if
end while
```

Figure 1: The generic search/learning algorithm.

More formally, following the LaSO framework, we assume that there is a set of input structures $\mathcal{X}$ and a set of output structures $\mathcal{Y}$ (in our case, elements $x \in \mathcal{X}$ will be documents and elements $y \in \mathcal{Y}$ will be documents marked up with mentions and their coreference sets). Additionally, we provide the structure of a search space $\mathcal{S}$ that results in elements of $\mathcal{Y}$ (we will discuss our choice for this component later in Section 3). The LaSO framework relies on a *monotonicity* assumption: given a structure $y \in \mathcal{Y}$ and a node n in the search space, we must be able to calculate whether it is *possible* for this node n to eventually lead to y (such nodes are called y-good).

LaSO parameterizes the search process with a weight vector $\boldsymbol{w} \in \mathbb{R}^D$, where weights correspond to features of search space nodes and inputs. Specifically, we write $\Phi : \mathcal{X} \times \mathcal{S} \to \mathbb{R}^D$ as a function that takes a pair of an input x and a node in the search space n and produces a vector of features. LaSO takes a standard search algorithm and modifies it to incorporate learning in an online manner to the algorithm shown in Figure 1. The key idea is to perform search as normal until a point at which it becomes impossible to reach the correct solution. When this happens, the weight vector $\boldsymbol{w}$ is updated in a corrective fashion. The algorithm relies on a parameter update formula; the two suggested by (Daumé III and Marcu, 2005) are a standard Perceptron-style update and an approximate large margin update of the sort proposed by (Gentile, 2001). In this work, we only use the large margin update, since in the original LaSO work, it consistently outperformed the simpler Perceptron updates. The update has the form given below:

$$\boldsymbol{w} \leftarrow \mathit{proj}\left(\boldsymbol{w} + Ck^{-1/2}\,\boldsymbol{\Delta}\right)$$

$$\boldsymbol{\Delta} = \mathit{proj}\left[\sum_{n \in sibs} \frac{\boldsymbol{\Phi}(x,n)}{|sibs|} - \sum_{n \in nodes} \frac{\boldsymbol{\Phi}(x,n)}{|nodes|}\right]$$

Where k is the update number, C is a tunable parameter and *proj* projects a vector into the unit sphere.

3 Joint EDT Model

The LaSO framework essentially requires us to specify two components: the search space (and corresponding operations) and the features. These two are inherently tied, since the features rely on the search space, but for the time being we will ignore the issue of the feature functions and focus on the search.

3.1 Search Space

We structure search in a left-to-right decoding framework: a hypothesis is a complete identification of the initial segment of a document. For instance, on a document with N words, a hypothesis that ends at position $0 < n < N$ is essentially what you would get if you took the full structured output and chopped it off at word n. In the example given in the introduction, one hypothesis might correspond to "Bill Clinton gave a" (which would be a y-good hypothesis), or to "Bill Clinton gave a" (which would not be a y-good hypothesis).

A hypothesis is expanded through the application of the search operations. In our case, the search procedure first chooses the number of words it is going to consume (for instance, to form the mention "Bill Clinton," it would need to consume two words). Then, it decides on an entity type and a mention type (or it opts to call this chunk not an entity (NAE), corresponding to non-underlined words). Finally, assuming it did not choose to form an NAE, it decides on which of the foregoing coreference chains this entity belongs to, or none (if it is the first mention of a new entity). All these decisions are made simultaneously, and the given hypothesis is then scored.

3.2 An Example

For concreteness, consider again the text given in the introduction. Suppose that we are at the word "them" and the hypothesis we are expanding is correct. That is, we have correctly identified "Bill Clinton" with entity type "person" and mention type "name;" that we have identified "the Senate" with entity type "organization" and mention type "name;" and that we have identified both "The President" and "his" as entities with entity type "person" and mention types "nominal" and "pronoun," respectively, and that "The President" points back to the chain ⟨Bill Clinton⟩ and that "his" points back to the chain ⟨Bill Clinton, The President⟩.

At this point of search, we have two choices for length: one or two (because there are only two words left: "them" and a period). A first hypothesis would be that the word "them" is NAE. A second hypothesis would be that "them" is a named person and is a new entity; a third hypothesis would be that "them" is a named person and is coreference with the "Bill Clinton" chain; a fourth hypothesis would be that "them" is a pronominal organization and is a new entity; next, "them" could be a pronominal organization that is coreferent with "the Senate"; and so on. Similar choices would be considered for the string "them ." when two words are selected.

3.3 Linkage Type

One significant issue that arises in the context of assigning a hypothesis to a coreference chain is how to compute features over that chain. As we will discuss in Section 4, the majority of our coreference-specific features are over *pairs* of chunks: the proposed new mention and an antecedent. However, since in general a proposed mention can have well more than one antecedent, we are left with a decision about how to combine this information.

The first, most obvious solution, is to essentially do nothing: simply compute the features over all pairs and add them up as usual. This method, however, intuitively has the potential for over-counting the effects of large chains. To compensate for this, one might advocate the use of an *average link* computation, where the score for a coreference chain is computed by averaging over its elements. One might also consider a *max link* or *min link* scenario, where one of the extrema is chosen as the value. Other research has suggested that a simple *last link*, where a mention is simply matched against the most recent mention in a chain might be appropriate, while *first link* might also be appropriate because the first mention of an entity tends to carry the most information.

In addition to these standard linkages, we also consider an *intelligent link* scenario, where the method of computing the link structure depends on the *mention type*. The intelligent link is computed as follow, based on the mention type of the current mention, m:

If m =**NAM then:** match *first* on NAM elements in the chain; if there are none, match against the *last* NOM element; otherwise, use *max link*.

If m =**NOM then:** match against the *max* NOM in the chain; otherwise, match against the most *last* NAM; otherwise, use *max link*.

If m =**PRO then:** use *average link* across all PRO or NAM; if there are none, use *max link*.

The construction of this methodology as guided by intuition (for instance, matching names against names is easy, and the first name tends to be the most complete) and subsequently tuned by experimentation on the development data. One might consider *learning* the best link method, and this may result in better performance, but we do not explore this option in this work. The initial results we present will be based on using intelligent link, but we will also compare the different linkage types explicitly.

4 Feature Functions

All the features we consider are of the form *base-feature* × *decision-feature*, where base features are functions of the input and decisions are functions of the hypothesis. For instance, a base feature might be something like "the current chunk contains the word 'Clinton"' and a decision feature might be something like "the current chunk is a named person."

4.1 Base Features

For pedagogical purposes and to facility model comparisons, we have separated the base features into eleven classes: lexical, syntactic, pattern-based, count-based, semantic, knowledge-based, class-based, list-based, inference-based, string match features and history-based features. We will deal with each of these in turn. Finally, we will discuss how these base features are combined into *meta-features* that are actually used for prediction.

Lexical features. The class of lexical features contains simply computable features of single words. This includes: the number of words in the current chunk; the unigrams (words) contained in this chunk; the bigrams; the two character prefixes and suffixes; the word stem; the case of the word, computed by regular expressions like those given by (Bikel et al., 1999); simple morphological features (number, person and tense when applicable); and, in the case of coreference, pairs of features between the current mention and an antecedent.

Syntactic features. The syntactic features are based on running an in-house state of the art part of speech tagger and syntactic chunker on the data. The words include unigrams and bigrams of part of speech as well as unigram chunk features. We have not used any parsing for this task.

Pattern-based features. We have included a whole slew of features based on lexical and part of speech patterns surrounding the current word. These include: eight hand-written patterns for identifying pleonastic "it" and "that" (as in "It is raining" or "It seems to be the case that ..."); identification of pluralization features on the previous and next head nouns (this is intended to help make decisions about entity types); the previous and next content verb (also intended to help with entity type identification); the possessor or possessee in the case of simple possessive constructions ("The president 's speech" would yield a feature of "president" on the word "speech", and vice-versa; this is indented to be a sort of weak sub-categorization principle); a similar feature but applied to the previous and next content verbs (again to provide a weak sort of subcategorization); and, for coreference, a list of part of speech and word sequence patterns that match up to four words between nearby mentions that are either highly indicative of coreference (e.g., "of," "said," "am" ", a") or highly indicative of non-coreference (e.g., "'s," "and," "in the," "and the"). This last set was generated by looking at intervening strings and finding the top twenty that had maximal mutual information with with class (coreferent or not coreferent) across the training data.

Count-based features. The count-based features apply only to the coreference task and attempt to capture regularities in the size and distribution of coreference chains. These include: the total number of entities detected thus far; the total number of mentions; the entity to mention ratio; the entity to word ratio; the mention to word ratio; the size of the hypothesized entity chain; the ratio of the number of mentions in the current entity chain to the total number of mentions; the number of intervening mentions between the current mention and the last one in our chain; the number of intervening mentions of the same type; the number of intervening sentence breaks; the Hobbs distance computed over syntactic chunks; and the "decayed density" of the hypothesized entity, which is computed as $\sum_{m=e} 0.5^{d(m)} / \sum_m 0.5^{d(m)}$, where m ranges over all previous mentions (constrained in the numerator to be in the same coreference chain as our mention) and $d(m)$ is the number of entities away this mention is. This feature is captures that some entities are referred to consistently across a document, while others are mentioned only for short segments, but it is relatively rare for an entity to be mentioned once at the beginning and then ignored again until the end.

Semantic features. The semantic features used are drawn from WordNet (Fellbaum, 1998). They include: the two most common synsets from WordNet for all the words in a chunk; all hypernyms of those synsets; for coreference, we also consider the distance in the WordNet graph between pairs of head words (defined to be the final word in the mention name) and whether one is a part of the other. Finally, we include the synset and hypernym information of the preceding and following verbs, again to model a sort of sub-categorization principle.

Knowledge-based features. Based on the hypothesis that many name to nominal coreference chains are best understood in terms of background knowledge (for instance, that "George W. Bush" is the "President"), we have attempted to take advantage of recent techniques from large scale data mining to extract lists of such pairs. In particular, we use the name/instance lists described by (Fleischman et al., 2003) and available on Fleischman's web page to generate features between names and nominals (this list contains $2m$ pairs mined from 15GBs of news data). Since this data set tends to focus mostly on person instances from news, we have additionally used similar data mined from a 138GB web corpus, for which more general "ISA" relations were mined (Ravichandran et al., 2005).

Class-based features. The class-based features we employ are designed to get around the sparsity of data problem while simultaneously providing new information about word usage. The first class-based feature we use is based on word classes derived from the web corpus mentioned earlier and computed as described by (Ravichandran et al., 2005). The second attempts to instill knowledge of collocations in the data; we use the technique described by (Dunning, 1993) to compute multi-word expressions and then mark words that are commonly used as such with a feature that expresses this fact.

List-based features. We have gathered a collection of about 40 lists of common places, organization, names, etc. These include the standard lists of names gathered from census data and baby name books, as well as standard gazetteer information listing countries, cities, islands, ports, provinces and states. We supplement these standard lists with lists of airport locations (gathered from the FAA) and company names (mined from the NASDAQ and NYSE web pages). We additionally include lists of semantically plural but syntactically singular words (e.g., "group") which were mined from a large corpus by looking for patterns such as ("members of the ..."). Finally, we use a list of persons, organizations and locations that were identified at least 100 times in a large corpus by the BBN IdentiFinder named entity tagger (Bikel et al., 1999).

These lists are used in three ways. First, we use simple list membership as a feature to improve detection performance. Second, for coreference, we look for word pairs that appear on the same list but are not identical (for instance, "Russia" and "England" appearing on the "country" list but not being identical hints that they are different entities). Finally, we look for pairs where one element in the pair is the head word from one mention and the other element in the pair is a list. This is intended to capture the notion that a word that appears on out "country list" is often coreferent with the word "country."

Inference-based features. The inference-based features are computed by attempting to infer an underlying semantic property of a given mention. In particular, we attempt to identify gender and semantic number (e.g., "group" is semantically plural although it is syntactically singular). To do so, we created a corpus of example mentions labels with number and gender, respectively. This data set was automatically extracted from our EDT data set by looking for words that corefer with pronouns for which we know the number or gender. For instance, a mention that corefers with "she" is known to be singular and female, while a mention that corefers with "they" is known to be plural. In about 5% of the cases, this was ambiguous – these cases were thrown out. We then used essentially the same features as described above to build a maximum entropy model for predicting number and gender. The predictions of this model are used both as features for detection as well as coreference (in the latter case, we check for matches). Additionally, we use several pre-existing classifiers as features. This are simple maximum entropy Markov models trained off of the MUC6 data, the MUC7 data and our ACE data.

String match features. We use the standard string match features that are described in every other coreference paper. These are: string match; substring match; string overlap; pronoun match; and normalized edit distance. In addition, we also use a string nationality match, which matches, for instance "Israel" and "Israeli," "Russia" and "Russian," "England" and "English," but not "Netherlands" and "Dutch." This is done by checking for common suffixes on nationalities and matching the first half of the of the words based on exact match. We additionally use a linguistically-motivated string edit distance, where the replacement costs are lower for vowels and other easily confusable characters. We also use the Jaro distance as an additional string distance metric. Finally, we attempt to match acronyms by looking at initial letters from the words in long chunks.

History-based features. Finally, for the detection phase of the task, we include features having to do with long-range dependencies between words. For instance, if at the beginning of the document we tagged the word "Arafat" as a person's name (perhaps because it followed "Mr." or "Palestinian leader"), and later in the document we again see the word "Arafat," we should be more likely to call this a person's name, again. Such features have previously been explored in the context of information extraction from meeting announcements using conditional random fields augmented with long-range links (Sutton and McCallum, 2004), but the LaSO framework makes no Markov assumption, so there is no extra effort required to include such features.

4.2 Decision Features

Our decision features are divided into three classes: simple, coreference and boundary features.

Simple. The simple decision features include: is this chunk tagged as an entity; what is its entity type; what is its entity subtype; what is its mention type; what is its entity type/mention type pair.

Coreference. The coreference decision features include: is this entity the start of a chain or continuing an existing chain; what is the entity type of this started (or continued) chain; what is the entity subtype of this started (or continued) chain; what is the mention type of this started chain; what is the mention type of this continued chain and the mention type of the most recent antecedent.

Boundary. The boundary decision features include: the second and third order Markov features over entity type, entity subtype and mention type; features appearing at the previous (and next) words within a window of three; the words that appear and the previous and next mention boundaries, specified also by entity type, entity subtype and mention type.

5 Experimental Results

5.1 Data

We use the official 2004 ACE training and test set for evaluation purposes; however, we exclude from the training set the Fisher conversations data, since this is very different from the other data sets and there is no Fisher data in the 2004 test set. This amounts to 392 training documents, consisting of $8.1k$ sentences and $160k$ words. There are a total of $24k$ mentions in the data corresponding to $10k$ entities (note that the data is not annotated for cross-document coreference, so instances of "Bill Clinton" appearing in two different documents are counted as two different entities). Roughly half of the entities are people, a fifth are organizations, a fifth are GPEs and the remaining are mostly locations or facilities. The test data is 192 documents, $3.5k$ sentences and $64k$ words, with $10k$ mentions to $4.5k$ entities. In all cases, we use a beam of 16 for training and test, and ignore features that occur fewer than five times in the training data.

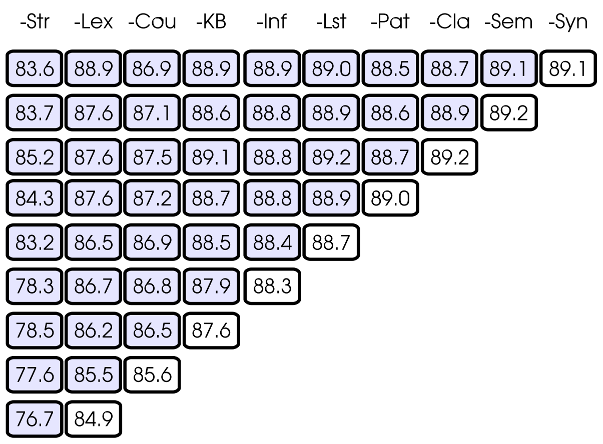
5.2 Evaluation Metrics

There are many evaluation metrics possible for this data. We will use as our primary measure of quality the ACE metric. This is computed, roughly, by first matching system mentions with reference mentions, then using those to match system entities with reference entities. There are costs, once this matching is complete, for type errors, false alarms and misses, which are combined together to give an ACE score, ranging from 0 to 100, with 100 being perfect (we use v.10 of the ACE evaluation script).

5.3 Joint versus Pipelined

We compare the performance of the joint system with the pipelined system. For the pipelined system, to build the mention detection module, we use the same technique as for the full system, but simply don't include in the hypotheses the coreference chain information (essentially treating each mention as if it were in its own chain). For the stand-alone coreference system, we assume that the correct mentions and types are always given, and simply hypothesize the chain (though still in a left-to-right manner).[1] Run as such, the joint model achieves an ACE score of 79.4 and the pipelined model achieves an ACE score of 78.1, a reasonably substantial improvement for performing both task simultaneously. We have also computed the performance of these two systems, ignoring the coreference scores (this is done by considering each mention to be its own entity and recomputing the ACE score). In this case, the joint model, ignoring its coreference output, achieves an ACE score of 85.6 and the pipelined model achieves a score of 85.3. The joint model does marginally better, but it is unlikely to be statistically significant. In the 2004 ACE evaluation, the best three performing systems achieved scores of 79.9, 79.7 and 78.2; it is unlikely that our system is significantly worse than these.

[1]One subtle difficulty with the joint model has to do with the online nature of the learning algorithm: at the beginning of training, the model is guessing randomly at what words are entities and what words are not entities. Because of the large number of initial errors made in this part of the task, the weights learned by the coreference model are initially very noisy. We experimented with two methods for compensating for this effect. The first was to give the mention identification model as "head start": it was run for one full pass through the training data, ignoring the coreference aspect and the following iterations were then trained jointly. The second method was to only update the coreference weights when the mention was identified correctly. On development data, the second was more efficient and outperformed the first by 0.6 ACE score, so we use this for the experiments reported in this section.



-Str	-Lex	-Cou	-KB	-Inf	-Lst	-Pat	-Cla	-Sem	-Syn
83.6	88.9	86.9	88.9	88.9	89.0	88.5	88.7	89.1	89.1
83.7	87.6	87.1	88.6	88.8	88.9	88.6	88.9	89.2	
85.2	87.6	87.5	89.1	88.8	89.2	88.7	89.2		
84.3	87.6	87.2	88.7	88.8	88.9	89.0			
83.2	86.5	86.9	88.5	88.4	88.7				
78.3	86.7	86.8	87.9	88.3					
78.5	86.2	86.5	87.6						
77.6	85.5	85.6							
76.7	84.9								

Figure 2: Comparison of performance as different feature classes are removed.

5.4 Feature Comparison for Coreference

In this section, we analyze the effects of the different base feature types on coreference performance. We use a model with perfect mentions, entity types and mention types (with the exception of pronouns: we do not assume we know pronoun types, since this gives away too much information), and measure the performance of the coreference system. When run with the full feature set, the model achieves an ACE score of 89.1 and when run with no added features beyond simple biases, it achieves 65.4. The best performing system in the 2004 ACE competition achieved a score of 91.5 on this task; the next best system scored 88.2, which puts us squarely in the middle of these two (though, likely not statistically significantly different). Moreover, the best performing system took advantage of additional data that they labeled in house.

To compute feature performance, we begin with all feature types and iteratively remove them one-by-one so that we get the best performance (we do not include the "history" features, since these are not relevant to the coreference task). The results are shown in Figure 2. Across the top line, we list the ten feature classes. The first row of results shows the performance of the system after removing just one feature class. In this case, removing lexical features reduces performance to 88.9, while removing string-match features reduces performance to 83.6. The non-shaded box (in this case, syntactic features) shows the feature set that can be removed with the least penalty in performance. The second row repeats this, after removing syntactic features.

As we can see from this figure, we can freely remove syntax, semantics and classes with little decrease in performance. From that point, patterns are dropped, followed by lists and inference, each with a performance drop of about 0.4 or 0.5. Removing the knowledge based features results in a large drop from 87.6 down to 85.6 and removing count-based features drops the performance another 0.7 points. Based on this, we can easily conclude that the most important feature classes to the coreference problem are, in order, string matching features, lexical features, count features and knowledge-based features, the latter two of which are novel to this work.

5.5 Linkage Types

As stated in the previous section, the coreference-only task with intelligent link achieves an ACE score of 89.1. The next best score is with min link (88.7) followed by average link with a score of 88.1. There is then a rather large drop with max link to 86.2, followed by another drop for last link to 83.5 and first link performs the poorest, scoring 81.5.

6 Discussion

In this paper, we have applied the *Learning as Search Optimization (LaSO)* framework to the entity detection and tracking task. The framework is an excellent choice for this problem, due to the fact that many relevant features for the coreference task (and even for the mention detection task) are highly non-local. This non-locality makes models like Markov networks intractable, and LaSO provides an excellent framework for tackling this problem. We have introduced a large set of new, useful features for this task, most specifically the use of knowledge-based features for helping with the name-to-nominal problem, which has led to a substantial improvement in performance. We have shown that performing joint learning for mention detection and coreference results in a better performing model that pipelined learning. We have also provided a comparison of the contributions of our various feature classes and compared different linkage types for coreference chains. In the process, we have developed an efficient model that is competitive with the best ACE systems.

Despite these successes, our model is not perfect: the largest source of error is with pronouns. This is masked by the fact that the ACE metric weights pronouns low, but a solution to the EDT problem should handle pronouns well. We intend to explore more complex features for resolving pronouns, and to incorporate these features into our current model. We also intend to explore more complex models for automatically extracting knowledge from data that can help with this task and applying this technique to a real application, such as summarization.

Acknowledgments: We thank three anonymous reviewers for helpful comments. This work was supported by DARPA-ITO grant NN66001-00-1-9814 and NSF grant IIS-0326276.

References

D. Bikel, R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34.

H. Daumé III and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*.

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

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

M. Fleischman, E. Hovy, and A. Echihabi. 2003. Offline strategies for online question answering: Answering questions before they are asked. In *ACL*.

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 *NAACL/HLT*.

C. Gentile. 2001. A new approximate maximal margin classification algorithm. *JMLR*, 2:213–242.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*.

D. Ravichandran, P. Pantel, and E. Hovy. 2005. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *ACL*.

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

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on Statistical Relational Learning*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *NIPS*.

Novelty Detection: The TREC Experience

Ian Soboroff and **Donna Harman**
National Institute of Standards and Technology
Gaithersburg, MD
`(ian.soboroff,donna.harman)@nist.gov`

Abstract

A challenge for search systems is to detect not only when an item is relevant to the user's information need, but also when it contains something new which the user has not seen before. In the TREC novelty track, the task was to highlight sentences containing relevant and new information in a short, topical document stream. This is analogous to highlighting key parts of a document for another person to read, and this kind of output can be useful as input to a summarization system. Search topics involved both news events and reported opinions on hot-button subjects. When people performed this task, they tended to select small blocks of consecutive sentences, whereas current systems identified many relevant and novel passages. We also found that opinions are much harder to track than events.

1 Introduction

The problem of novelty detection has long been a significant one for retrieval systems. The "selective dissemination of information" (SDI) paradigm assumed that the people wanted to be able to track new information relating to known topics as their primary search task. While most SDI and information filtering systems have focused on similarity to a topical profile (Robertson, 2002) or to a community of users with a shared interest (Belkin and Croft, 1992), recent efforts (Carbonell and Goldstein, 1998; Allan et al., 2000; Kumaran et al., 2003) have looked at the retrieval of specifically *novel* information.

The TREC novelty track experiments were conducted from 2002 to 2004 (Harman, 2002; Soboroff and Harman, 2003; Soboroff, 2004). The basic task was defined as follows: given a topic and an ordered set of documents related to that topic, segmented into sentences, return those sentences that are both relevant to the topic and novel given what has already been seen previously in that document set. This task models an application where a user is skimming a set of documents, and the system highlights new, on-topic information.

There are two problems that participants must solve in this task. The first is identifying relevant sentences, which is essentially a passage retrieval task. Sentence retrieval differs from document retrieval because there is much less text to work with, and identifying a relevant sentence may involve examining the sentence in the context of those surrounding it. The sentence was specified as the unit of retrieval in order to standardize the task across a variety of passage retrieval approaches, as well as to simplify the evaluation.

The second problem is that of identifying those relevant sentences that contain new information. The operational definition of "new" here is information that has not appeared previously in this topic's set of documents. In other words, we allow the system to assume that the user is most concerned about finding new information in this particular set of documents, and is tolerant of reading information he already knows because of his background knowledge. Since each sentence adds to the user's knowledge, and later sentences are to be retrieved only if they contain new information, novelty retrieval resembles a filtering task.

Novelty is an inherently difficult phenomenon to operationalize. Document-level novelty detection, while intuitive, is rarely useful because nearly every document contains something new, particularly when the domain is news. Hence, our decision to use sentences as the unit of retrieval. Moreover, determining ground truth for a novelty detection task is more difficult than for topical relevance, because one is forced not only to face the idiosyncratic nature of relevance, but also to rely all the more on the memory and organizational skills of the assessor, who must try and remember everything he has read. We wanted to determine if people could accomplish this task to any reasonable level of agreement, as well as to see what computational approaches best solve this problem.

2 Input Data

The first year of the novelty track (Harman, 2002) was a trial run in several ways. First, this was a new task for the community and participating groups had no training data or experience. But second, it was unclear how humans would perform this task and therefore creating the "truth" data was in itself a large experiment. NIST decided to minimize the cost by using 50 old topics from TRECs 6, 7, and 8.

The truth data was created by asking NIST assessors (the humans performing this task) to identify the set of relevant sentences from each relevant document and then from that set of relevant sentences, mark those that were novel. Specifically, the assessors were instructed to identify a list of sentences that were:

1. relevant to the question or request made in the description section of the topic,
2. their relevance was independent of any surrounding sentences,
3. they provided new information that had not been found in any previously picked sentences.

Most of the NIST assessors who worked on this task were not the ones who created the original topics, nor had they selected the relevant documents. This turned out to be a major problem. The assessors' judgments for the topics were remarkable in that only a median of 2% of the sentences were judged to be relevant, despite the documents themselves being relevant. As a consequence, nearly every relevant sentence (median of 93%) was declared novel. This was due in large part to assessor disagreement as to relevancy, but also that fact that this was a new task to the assessors. Additionally, there was an encouragement not to select consecutive sentences, because the goal was to identify relevant and novel sentences minimally, rather than to try and capture coherent blocks of text which could stand alone. Unfortunately, this last instruction only served to confuse the assessors. Data from 2002 has not been included in the rest of this paper, nor are groups encouraged to use that data for further experiments because of these problems.

In the second year of the novelty track (Soboroff and Harman, 2003), the assessors created their own new topics on the AQUAINT collection of three contemporaneous newswires. For each topic, the assessor composed the topic and selected twenty-five relevant documents by searching the collection. Once selected, the documents were ordered chronologically, and the assessor marked the relevant sentences and those relevant sentences that were novel. No instruction or limitation was given to the assessors concerning selection of consecutive sentences, although they were told that they did not need to choose an otherwise irrelevant sentence in order to resolve a pronoun reference in a relevant sentence. Each topic was independently judged by two different assessors, the topic author and a "secondary" assessor, so that the effects of different human judgments could be measured. The judgments of the primary assessor were used as ground truth for evaluation, and the secondary assessor's judgments were taken to represent a ceiling for system performance in this task.

Another new feature of the 2003 data set was a division of the topics into two types. Twenty-eight of the fifty topics concerned events such as the bombing at the 1996 Olympics in Atlanta, while the remaining topics focused on opinions about controversial subjects such as cloning, gun control, and same-sex marriages. The topic type was indicated in the topic description by a `<toptype>` tag.

This pattern was repeated for TREC 2004 (Soboroff, 2004), with fifty new topics (twenty-five events and twenty-five opinion) created in a similar manner and with the same document collection. For 2004, assessors also labeled some documents as irrelevant, and irrelevant documents up through the first twenty-five relevant documents were included in the document sets distributed to the participants. These irrelevant documents were included to increase the "noise" in the data set. However, the assessors only judged sentences in the relevant documents, since, by the TREC standard of relevance, a document is considered relevant if it contains any relevant information.

3 Task Definition

There were four tasks in the novelty track:

Task 1. Given the set of documents for the topic, identify all relevant and novel sentences.

Task 2. Given the relevant sentences in all documents, identify all novel sentences.

Task 3. Given the relevant and novel sentences in the first 5 documents **only**, find the relevant and novel sentences in the remaining documents. Note that since some documents are irrelevant, there *may not be* any relevant or novel sentences in the first 5 documents for some topics.

Task 4. Given the relevant sentences from all documents, and the novel sentences from the first 5 documents, find the novel sentences in the remaining documents.

These four tasks allowed the participants to test their approaches to novelty detection given different levels of training: none, partial, or complete relevance information, and none or partial novelty information.

The test data for a topic consisted of the topic statement, the set of sentence-segmented documents, and the chronological order for those documents. For tasks 2-4, training data in the form of relevant and novel "sentence qrels" were also given. The data was released and results were submitted in stages to limit "leakage" of training data between tasks. Depending on the task, the system was to output the identifiers of sentences which the system determined to contain relevant and/or novel relevant information.

4 Evaluation

Because novelty track runs report their relevant and novel sentences as an unranked set, traditional measures of ranked retrieval effectiveness such as mean average precision can't be used. One alternative is to use set-based recall and precision. Let M be the number of matched sentences, i.e., the number of sentences selected by both the assessor and the system, A be the number of sentences selected by the assessor, and S be the number of sentences selected by the system. Then sentence set recall is $R = M/A$ and precision is $P = M/S$.

However, set-based recall and precision do not average well, especially when the assessor set sizes A vary widely across topics. Consider the following example as an illustration of the problems. One topic has hundreds of relevant sentences and the system retrieves 1 relevant sentence. The second topic has 1 relevant sentence and the system retrieves hundreds of sentences. The average for both recall and precision over these two topics is approximately .5 (the scores on the first topic are 1.0 for precision and essentially 0.0 for recall, and the scores for the second topic are the reverse), even though the system did precisely the wrong thing. While most real systems wouldn't exhibit this extreme behavior, the fact remains that set recall and set precision averaged over a set of topics is not a robust diagnostic indicator

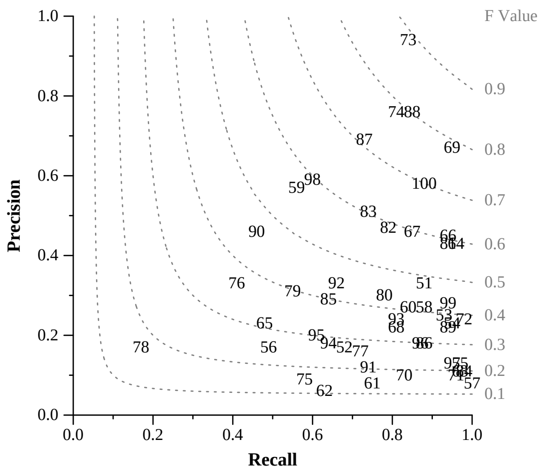


Figure 1: The F measure, plotted according to its precision and recall components. The lines show contours at intervals of 0.1 points of F. The black numbers are per-topic scores for one TREC system.

of system performance. There is also the problem of how to define precision when the system returns no sentences ($S = 0$). Leaving that topic out of the evaluation for that run would mean that different systems would be evaluated over different numbers of topics. The standard procedure is to define precision to be 0 when $S = 0$.

To avoid these problems, the primary measure used in the novelty track was the F measure. The F measure (which is itself derived from van Rijsbergen's E measure (van Rijsbergen, 1979)) is a function of set recall and precision, together with a parameter β which determines the relative importance of recall and precision:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

A β value of 1, indicating equal weight, is used in the novelty track:

$$\mathrm{F}_{\beta=1} = \frac{2PR}{P + R}$$

Alternatively, this can be formulated as

$$\mathrm{F}_{\beta=1} = \frac{2 \times (\text{\# relevant retrieved})}{(\text{\# retrieved}) + (\text{\# relevant})}$$

For any choice of β, F lies in the range $[0, 1]$, and the average of the F measure is meaningful even when the judgment sets sizes vary widely. For example, the F measure in the scenario above is essentially 0, an intuitively appropriate score for such behavior. Using the F measure also deals with the problem of what to do when the system returns no sentences since recall is 0 and the F measure is legitimately 0 regardless of what precision is defined to be.

Note, however, that two runs with equal F scores do not indicate equal precision and recall. The contour lines in Figure 1 illustrate the shape of the F measure in recall-precision space. An F score of 0.5, for example, can describe a range of precision and recall scores. Figure 1 also shows the per-topic scores for a particular TREC run. It is easy to see that topics 98, 83, 82, and 67 exhibit a wide range of performance, but all have an F score of close to 0.6. Thus, two runs with equal F scores may be performing quite differently, and a difference in F scores can be due to changes in precision, recall, or both. In practice, if F is used, precision and recall should also be examined, and we do so in the analysis which follows.

5 Analysis

5.1 Analysis of truth data

Since the novelty task requires systems to automatically select the same sentences that were selected manually by the assessors, it is important to analyze the characteristics of the manually-created truth data in order to better understand the system results. Note that the novelty task is both a passage retrieval task, i.e., retrieve relevant sentences, and a novelty task, i.e., retrieve only relevant sentences that contain new information.

In terms of the passage retrieval part, the TREC novelty track was the first major investigation into how users select relevant parts of documents. This leads to several obvious questions, such as what percentage of the sentences are selected as relevant, and do these sentences tend to be adjacent/consecutive? Additionally, what kinds of variation appear, both across users and across topics. Table 1 shows the median percentage of sentences that were selected as relevant, and what percentage of these sentences were consecutive. Since each topic was judged by two assessors, it also shows the percentage of sentences selected by assessor 1 (the "official" assessor used in scoring) that were also selected by assessor 2. The table gives these percentages for all topics and also broken out into the two types of topics (events and opinions).

First, the table shows a large variation across the two years. The group in 2003 selected more relevant sentences (almost 40% of the sentences were selected as relevant), and in particular selected many consecutive sentences (over 90% of the relevant sentences were adjacent). The median length of a string of consecutive sentences was 2; the mean was 4.252 sentences. The following year, a different group of assessors selected only about half as many relevant sentences (20%), with fewer consecutive sentences. This variation across years may reflect the group of assessors in that the 2004 set were TREC "veterans" and were more likely to be very selective in terms of what was considered relevant.

The table also shows a variation across topics, in particular between topics asking about events versus those asking about opinions. The event topics, for both years, had more relevant sentences, and more consecutive sentences (this effect is more apparent in 2004).

Agreement between assessors on which sentences were relevant was fairly close to what is seen in document relevance tasks. There was slightly more agreement in 2003, but there were also many more relevant sentences so the likelihood of a match was higher. There is more agreement on events than on opinions, partially for the same reason, but also because there is generally less agreement on what constitutes an opinion. These medians hide a wide range of judging behavior across the assessors, particularly in 2003.

The final two rows of data in the table show the medians for novelty. There are similar patterns to those seen in the relevant sentence data, with the 2003 assessors clearly being more liberal in judging. However, the pattern is reversed for topic types, with more sentences being considered relevant and novel for the opinion topics than for the event topics. The agreement on novelty is less than on relevance, particularly in 2004 where there were smaller numbers of novel and relevant sentences selected.

Another way to look at agreement is with the kappa statistic (Cohen, 1960). Kappa computes whether two assessors disagree, with a correction for "chance agreement" which we would expect to occur randomly. Kappa is often interpreted as the degree of agreement between assessors, although this interpretation is not well-defined and varies from field to field (Di Eugenio, 2000). For relevant sentences across all topics in the 2004 data set, the kappa value is 0.549, indicating statistically significant agreement between the assessors but a rather low-to-moderate degree of agreement by most scales of interpretation. Given that agreement is usually not very high for relevance judgments (Voorhees, 1998), this is as expected.

5.2 Analysis of participants results

Most groups participating in the 2004 novelty track employed a common approach, namely to measure relevance as similarity to the topic and novelty as

		2003	2004
Relevant	all topics	0.39	0.20
	events only	0.47	0.25
	opinions only	0.38	0.15
Consecutive	all topics	0.91	0.70
	events only	0.93	0.85
	opinions only	0.91	0.65
Relevant agreement	all topics	0.69	0.60
	events only	0.82	0.68
	opinions only	0.63	0.50
Novelty	all topics	0.68	0.40
	events only	0.61	0.38
	opinions only	0.73	0.42
Novelty agreement	all topics	0.56	0.35
	events only	0.65	0.45
	opinions only	0.48	0.29

Table 1: Median fraction of sentences which were relevant and novel, fraction of consecutive relevant sentences, and proportion of agreement by the secondary assessor.

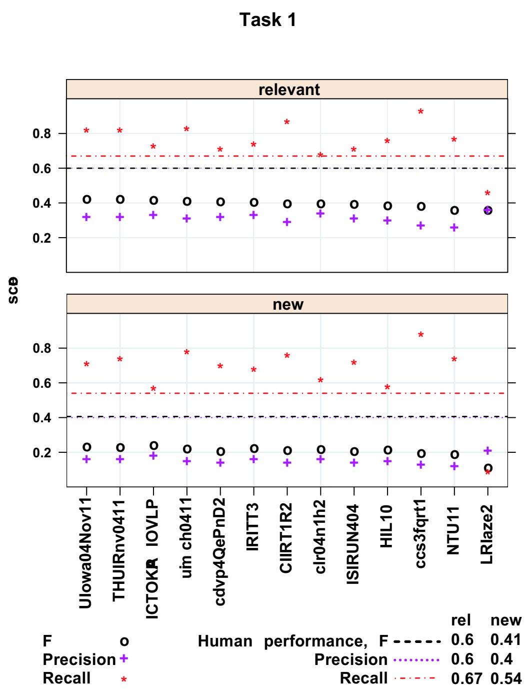


Figure 2: Task 1 precision, recall, and F scores for the top run from each group in TREC 2004

dissimilarity to past sentences. On top of this framework the participants used a wide assortment of methods which may be broadly categorized into statistical and linguistic methods. Statistical methods included using traditional retrieval models such as tf.idf and Okapi coupled with a threshold for retrieving a relevant or novel sentence, expansion of the topic and/or document sentences using dictionaries or corpus-based methods, and using named entities as features. Some groups also used machine learning algorithms such as SVMs in parts of their detection process. Semantic methods included deep parsing, matching discourse entities, looking for particular verbs and verb phrases in opinion topics, coreference resolution, normalization of named entities, and in one case manual construction of ontology's for topic-specific concepts.

Figure 2 shows the Task 1 results for the top run from each group in TREC 2004. Groups employing statistical approaches include UIowa, CIIR, UMich, and CDVP. Groups employing more linguistic methods include CLR, CCS, and LRI. THU and ICT took a sort of kitchen-sink approach where each of their runs in each task tried different techniques, mostly statistical.

The F scores for both relevance and novelty retrieval are fairly uniform, and they are dominated by the precision component. The top scoring systems by F score are largely statistical in nature; for example, see (Abdul-Jaleel et al., 2004) (CIIR) and (Eichmann et al., 2004) (UIowa). CLR (Litkowski, 2004) and LRI (Amrani et al., 2004), which use much stronger linguistic processing, achieve the highest precision at the expense of recall. Overall, precision is quite low and recall is high, implying that most systems are erring in favor of retrieving many sentences.

A closer comparison of the runs among themselves and to the truth data confirms this hypothesis. While the 2004 assessors were rather selective in choosing relevant and novel sentences, often selecting just a handful of sentences from each document, the systems were not. The systems retrieved an average of 49.5% of all sentences per topic as relevant, compared to 19.2% chosen by the assessor. Furthermore, the runs chose 41% of all sentences (79% of their own relevant sentences) as novel, compared to the assessor who selected only 8.4%. While these numbers are a very coarse average that ignores differences between the topics and between the documents in each set, it is a fair summary of the data. Most of the systems called nearly every sentence relevant and novel. By comparison, the person attempting this task (the second assessor, scored as a run and shown as horizontal lines in Figure 2) was much more effective than the systems.

The lowest scoring run in this set, LRIaze2, actually has the highest precision for both relevant and novel sentences. The linguistics-driven approach of this group included standardizing acronyms, building a named-entity lexicon, deep parsing, resolving coreferences, and matching concepts to manually-built, topic-specific ontologies (Amrani et al., 2004). A close examination of this run's pattern shows that they retrieved very few sentences, in line with the amounts chosen by the assessor. They were not often the correct sentences, which accounts for the low recall, but by not retrieving too many false alarms, they managed to achieve a high precision.

Our hypothesis here is that the statistical systems, which are essentially using algorithms designed for document retrieval, approached the sentences with an overly-broad term model. The majority of the documents in the data set are relevant, and so many of the topic terms are present throughout the documents. However, the assessor was often looking for a finer-grained level of information than what exists at the document level. For example, topic N51 is concerned with Augusto Pinochet's arrest in London. High-quality content terms such as Pinochet, Chile, dictator, torture, etc appear in nearly every sentence, but the key relevant ones — which are very few — are those which specifically talk about the arrest. Most systems flagged nearly every sentence as relevant, when the topic was much narrower than the documents themselves.

One explanation for this may be in how thresholds were learned for this task. Since task 1 provides no data beyond the topic statement and the documents themselves, it is possible that systems were tuned to the 2003 data set where there are more relevant sentences. However, this isn't the whole story, since the difference in relevant sentences between 2003 and 2004 is not so huge that it can explain the rates of retrieval seen here. Additionally, in task 3 some topic-specific training data was provided, and yet the effectiveness of the systems was essentially the same.

Of those systems that tried a more fine-grained approach, it appears that it is complicated to learn exactly which sentences contain the relevant information. For example, nearly every system had trouble identifying relevant opinion sentences. One might expect that those systems which analyzed sentence structure more closely would have done better here, but there is essentially no difference. Identifying relevant information at the sentence level is a very hard problem.

We see very similar results for novel sentence retrieval. Rather than looking at task 1, where systems retrieved novel from their own selection of relevant sentences, it's better to look at runs in task 2 (Figure 3). Since in this task the systems are given all relevant sentences and just search for novelty, the baseline performance for comparison is just labeling all the sentences as novel. Most systems, surprisingly including the LRI run, essentially do retrieve nearly every sentence as novel. The horizontal lines show the baseline performance; the baseline recall is 1.0 and is at the top of the Y axis. All the runs except clr04n2 are just above this baseline, with cdvp4NTerFr1 and novcolrcl the most discriminating.

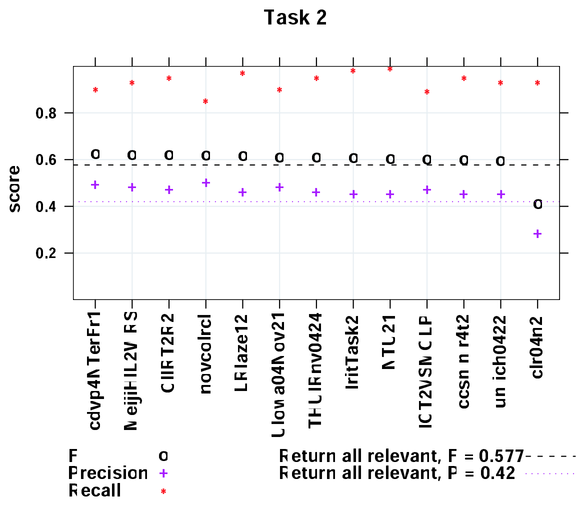


Figure 3: Task 2 scores for the top run from each group in TREC 2004

The approach of Dublin City University (cdvp4NTerFr1) is essentially to set a threshold on the tf.idf value of the unique words in the given sentence, but their other methods which incorporate the history of unique terms and the difference in sentence frequencies between the current and past sentences perform comparably (Blott et al., 2004). Similarly, Columbia University (novcolrcl) focuses on previously unseen words in the current sentence as the main evidence of novelty (Schiffman and McKeown, 2004). As opposed to the ad hoc threshold in the DCU system, Columbia employs a hill-climbing approach to learning the threshold. This particular run is optimized for recall; another optimized for precision achieved the highest precision of all task 2 runs, but with very low recall. In general, we conclude that most systems achieving high scores in novelty detection are recall-oriented and as a result still provide the user with too much information.

As was mentioned above, opinion topics proved much harder than events. Every system but one did better on event topics than on opinions in task 1

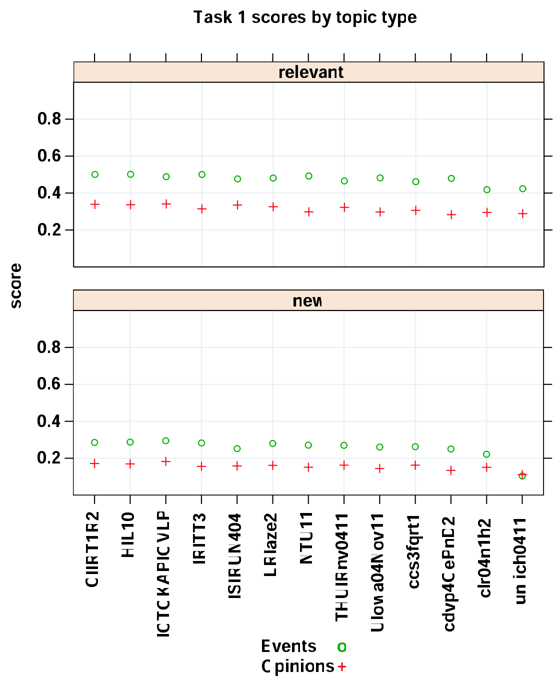


Figure 4: F scores for event and opinion topics in task 1.

(Figure 4). In task 2, where all relevant sentences were provided, many runs do as well or better on opinion topics than events. Thus, the complexity for opinions is more in finding which sentences contain them, than determining which opinions are novel.

6 Conclusion

The novelty track in TREC examined a particular kind of novelty detection, that is, finding novel, on-topic sentences within documents that the user is reading. Both statistical and linguistic techniques, as well as filtering and learning approaches can be applied to detecting novel relevant information within documents, but nevertheless it is a hard problem for several reasons. First, because the unit of interest is a sentence, there is not a lot of data in each unit on which to base the decision. When the document as a whole is relevant, techniques designed for document retrieval seem unable to make fine distinctions about which sentences within the document contain the relevant information. Initial threshold setting is critical and difficult.

When we examined human performance on this task, it is clear that users do make very fine distinctions. Looking particularly at the 2004 set of relevant and novel sentences, less than 20% of the sentences in relevant documents were marked as relevant, and only 40% of those (or 8% of the total sentences) were marked as both relevant and novel.

The TREC novelty data sets themselves support some interesting uses outside of the novelty track. Whereas the data from 2002 is clearly flawed and should not be used, the data from 2003 and 2004 can be regarded as valid samples of user input in terms of relevant sentence selection, and further reduction of those sentences to those presenting new information. One obvious use is in the passage retrieval arena, e.g., using the relevant sentences for testing passage retrieval, either at the single sentence level or using the consecutive sentences to test when to retrieve multiple sentences. A second use is for summarization, where the relevant AND novel sentences can serve as the truth data for the extraction phase (and then compressed in some manner). Other uses of the data include manual analysis of user behavior when processing documents in response to a question, or looking further into the user agreement issues, particularly in the summarization area.

The novelty data is also unique in that it deliberately contains a mix of topics on events and on opinions regarding controversial subjects. The opinions topics are quite different in this regard than other TREC topics, which have historically focused on events or narrative information on a subject or person. This exploration has been an interesting and fruitful one. By mixing the two topic types within each task, we see that identifying opinions within documents is hard, even with training data, while detecting new opinions (given relevance) seems analogous to detecting new information about an event.

References

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, `http://trec.nist.gov/pub/trec13/papers/umass.novelty.hard.pdf`.

James Allan, Victor Lavrenko, and Hubert Jin. 2000. First story detection in TDT is hard. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2000)*, pages 374–381.

Ahmed Amrani, Jérôme Azé, Thomas Heitz, Yves Kodratoff, and Mathieu Roche. 2004. From the texts to the concepts they contain: a chain of linguistic treatments. In *Proceedings of the Thirteenth Text REtrieval Confer-*

ence (TREC 2004), http://trec.nist.gov/pub/trec13/papers/uparis.novelty2.pdf.

Nicholas J. Belkin and W. Bruce Croft. 1992. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December.

Stephen Blott, Oisin Boydell, Fabrice Camous, Paul Ferguson, Georgina Gauhan, Cathal Gurrin, Gareth J. F. Jones, Noel Murphy, Noel O'Connor, Alan F. Smeaton, Barry Smyth, and Peter Wilkins. 2004. Experiments in terabyte searching, genomic retrieval and novelty detection for TREC-2004. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, http://trec.nist.gov/pub/trec13/papers/dcu.tera.geo.novelty.pdf.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval (SIGIR '98)*, Melbourne, Australia, August, pages 335–336. ACM Press.

J. A. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Barbara Di Eugenio. 2000. On the usage of kappa to evaluate agreement on coding tasks. In *Proceedings of the Second International Conference on Language Resources and Evaluatation (LREC 2000)*, Athens, Greece, pages 441–446.

David Eichmann, Yi Zhang, Shannon Bradshaw, Xin Ying Qui, Li Zhou, Padmini Srinivasan, Aditya Kumar Sehgal, and Hudon Wong. 2004. Novelty, question answering and genomics: the University of Iowa response. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, http://trec.nist.gov/pub/trec13/papers/uiowa.novelty.qa.geo.pdf.

Donna Harman. 2002. Overview of the TREC 2002 novelty track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, NIST Special Publication 500-251, pages 46–55, Gaithersburg, MD, November.

Girindhar Kumaran, James Allan, and Andrew McCallum. 2003. Classification models for new event detection. Technical Report IR-362, CIIR, University of Massachusetts, Amherst.

Kenneth C. Litkowski. 2004. Evolving XML and dictionary strategies for question answering and novelty tasks. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, http://trec.nist.gov/pub/trec13/papers/clresearch.qa.novelty.pdf.

Stephen E. Robertson. 2002. Introduction to the special issue: Overview of the TREC routing and filtering tasks. *Information Retrieval*, 5:127–137.

Barry Schiffman and Kathleen R. McKeown. 2004. Columbia university in the novelty track at TREC 2004. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, http://trec.nist.gov/pub/trec13/papers/columbiau.novelty.pdf.

Ian Soboroff and Donna Harman. 2003. Overview of the TREC 2003 novelty track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, NIST Special Publication 500-255, Gaithersburg, MD, November.

Ian Soboroff. 2004. Overview of the TREC 2004 novelty track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, http://trec.nist.gov/pub/trec13/papers/NOVELTY.OVERVIEW.pdf.

C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths.

Ellen M. Voorhees. 1998. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval (SIGIR '98)*, Melbourne, Australia, August, pages 315–323. ACM Press.

Tell Me What You Do and I'll Tell You What You Are: Learning Occupation-Related Activities for Biographies

Elena Filatova[*]
Department of Computer Science
Columbia University
New York, NY
filatova@cs.columbia.edu

John Prager
IBM T.J. Watson
Research Center
Yorktown Heights, NY
jprager@us.ibm.com

Abstract

Biography creation requires the identification of important events in the life of the individual in question. While there are events such as birth and death that apply to everyone, most of the other activities tend to be occupation-specific. Hence, occupation gives important clues as to which activities should be included in the biography. We present techniques for automatically identifying which *important* events apply to the general population, which ones are occupation-specific, and which ones are person-specific. We use the extracted information as features for a multi-class SVM classifier, which is then used to automatically identify the occupation of a previously unseen individual. We present experiments involving 189 individuals from ten occupations, and we show that our approach accurately identifies general and occupation-specific activities and assigns unseen individuals to the correct occupations. Finally, we present evidence that our technique can lead to efficient and effective biography generation relying only on statistical techniques.

1 Introduction

Natural language processing (NLP) applications such as summarization and question-answering (QA) systems are designed to reduce the amount of time necessary for finding information of interest. Summarization systems produce a condensed version of the generally important information presented in the input, while QA systems target specific information according to a certain question.

Recently there has been increased interest in creating systems which combine summarization and QA and can give long answers to definition, biography, opinion and other question types. Such systems use general summarization techniques and at the same time take advantage of the fact that the selected information is not *generally* important but should be targeted towards answering the user's request. This idea is exploited in DUC 2004,[1] where one of the tasks is to create a summary targeting an answer to the "Who is X?" question. The systems that showed the highest performance for this task, combine traditional summarization techniques and also have modules developed specifically for creating summaries containing biographical information. Using a set of biographical facts proved to be useful to answer questions other than "What is X?" (Prager et al., 2004).

To extract biographical facts it is useful to understand the nature of different human activities. Biographical activities such as birth, death, living somewhere are applicable to all people, while each occupation is associated with its own set of activities.

In this work we suggest a novel unsupervised approach for automatic extraction of general biographical activities and activities typical for people of a particular occupation. To extract such activities we use *atomic events* (Filatova and Hatzivassiloglou, 2003) and statistical algorithms based on Markov Chains. Before creating a biography of a person as a representative of some occupation it is necessary to find out the occupation of this person; to do this we use SVM classification with activities as features.

In Section 2 we describe the current approaches for biography creation. In Section 3 we give an overview of atomic events and formulate our approach towards extracting occupation-specific activities. In Section 4 we describe mathematical models we use to identify activities typical for different occupations. In Section 5 we describe an SVM classification procedure for assigning people to the appropriate occupations. Finally, Section 6 summarizes the results and suggests a plan for future work.

[*]Part of this work was conducted while Elena Filatova was a summer intern at the IBM T. J. Watson Research Center.

[1]Document Understanding Conferences are a testbed for evaluating summarization systems

2 Related work

The systems participating in DUC 2004 create summaries which could be used as answers to the question "Who is X?". These systems use a wide variety of techniques. Blair-Goldensohn et al. (2004a) treat "Who is X?" as a definition question and use the DefScriber system (Blair-Goldensohn et al., 2004b). Biryukov et al. (2005) use Topic Signatures (Lin and Hovy, 2000) constructed around the person's name. Zhou et al. (2004) use nine features which are likely to be used in biography texts: bio (biographical facts), fame, personality, social, education, nationality, scandal, personal, work. Using manual annotation of 130 biographies they learn the textual patterns corresponding to these nine features.

Biographical information can be used to answer not only "Who is X?" questions. Prager et al. (2004) use biographical information within their QA-by-Dossier-with-Constraints system, which checks whether the possible answer satisfies the constraints for the person about whom the question is asked. For example, a natural constraint for artists, composers and writers is that all their works are produced in the span of time between the dates of birth and death.

Biographies vary greatly in length, genre and the presented information. Biographies of the same person in encyclopedias and yellow press magazines might contain different information. Encyclopedic biographies contain dates of birth and death, the most important achievements of people, while yellow press magazines tend to describe less important, usually scandalous facts of someone's life.

In this work we assume that most biographies can be broken into two main parts: biographical facts (the person's place and date of birth, where the person lived); and activities typically associated with the person's occupation (e.g., singers sing, explorers travel to study new lands, artists create paintings). Existing research shows that knowing the persons's occupation is helpful for detecting information which should be used in the biography (Schiffman et al., 2001; Duboue and McKeown, 2003).

3 Automatic extraction of activities typical for different occupations

3.1 Data

We created our own set of people belonging to various occupations as we were aware of no set diverse enough to analyze activities of people belonging to different occupations. We could not use the list of people whose biographies were created for DUC 2004 task: as the input documents for this task were contemporary newswire articles, more than a half of the 50 people used there were politicians.

We therefore performed what might be considered a pilot study. We chose 10 occupations and 20 practitioners of each. We understood that 10 occupations would not cover every person mentioned in a news corpus, but that was not critical to our study.

As described later, we sought documents for each chosen person. Since no documents were found for some of the individuals, these people were eliminated from the experiments; 189 survived. We ended up with the following sets of collections:

a. 20 artists	f. 20 mathematicians
b. 18 athletes	g. 19 physicists
c. 20 composers	h. 20 politicians
d. 15 dancers	i. 20 singers
e. 17 explorers	j. 20 writers

We found that for some occupations human annotators agree upon its representatives however different they are. For example, to whatever school an artist belongs (impressionism, surrealism) he/she is usually addressed as an *artist*. The situation with *politicians* is different. They are often referred to not as politicians but according to the post held (president, prime-minister). Choosing an appropriate occupation title becomes crucial at the document retrieval stage as this title is used to query the search engine.

Our goals for the occupation list are that it satisfies the following criteria:

- it is diverse and covers a substantial variety of occupations from arts, sciences and other aspects of human activities;
- it contains some occupations which are closely related between each other and might be later merged into one superclass, for example, mathematicians and physicists;
- it contains occupations that are very different and it is almost impossible to specify activities which are routinely performed in two occupations, for example, singers and explorers.

To get the lists of people belonging to each particular occupation we use WordNet 2.0 (Fellbaum, 1998) (e.g., hyponyms for *composer* contain a list of composers). We also use "Google Sets" interface,[2] it was previously successfully used to find people belonging to the same occupation (Prager et al., 2004).

We retrieve documents from four corpora: AQUAINT, TREC, part of World Book and part of

[2]`http://labs.google.com/sets`

Encyclopedia Britannica. For document retrieval we use IBM's JuruXML search engine (Carmel et al., 2001) which allows one to index terms along with any associated named entity class labels. Queries to JuruXML may include tagged terms, which will only match similarly tagged instances in the index. We use ⟨person⟩ and ⟨role⟩ tags in the queries to perform word sense disambiguation of two types: to differentiate a person from, for example, a location with the same name (e.g., Newton - a physicist and Newton - a town in Massachusetts); and to differentiate two different people with the same name belonging to different occupations (e.g., Louis Armstrong a singer and Lance Armstrong an athlete). The second issue can be partially avoided by submitting full name of a person but it reduces the amount of documents retrieved about this person dramatically. Thus, we retrieve all the documents about people by submitting the query "*⟨person⟩Name⟨/person⟩ ⟨role⟩Occupation⟨/role⟩*".

The number of documents retrieved varied from one, for the query "*⟨person⟩Cauchy⟨/person⟩ ⟨role⟩mathematician⟨/role⟩*," to up to 8,144, for "*⟨person⟩Clinton⟨/person⟩ ⟨role⟩politician⟨/role⟩*." To counteract misbalance in the data we relied on the $tf.idf$ ranking of JuruXML to sort the matching documents. The top ten such documents were kept (or all of them if fewer than ten were returned).

3.2 Extracting occupation-specific activities

To automatically discover general and occupation-specific activities we use a modified version of atomic events (Filatova and Hatzivassiloglou, 2003). Atomic events are triplets consisting of two named entities and a verb which labels the relation between these two named entities. We extract 189 lists of atomic events according to the following procedure:

1. For each person analyze the corresponding collection of documents retrieved for this person.
2. From every sentence containing the name of the person under analysis extract all the pairs of named entities, one of the elements of which is the name of this person.
3. For every such pair of named entities extract all verbs, excluding modal and auxiliary verbs, that appear in-between them.
4. Count how many times each triplet containing two named entities and a verb in-between appears in the collection of documents describing the person under analysis.

First Named Entity	Verb	Second Named Entity
Columbus/PERSON	died/VBN	1506/DATE
Columbus/PERSON	sailed/VBD	India/PLACE

Table 1: A sample of atomic events extracted for the collection of documents about Christopher Columbus

First Named Entity	Verb	Second Named Entity
Vespucci/PERSON	explored/VBD	S. America/PLACE
Bering/PERSON	explored/VBD	Aleutian/PLACE

Table 2: A sample of atomic events extracted for the collection of documents about Christopher Columbus

The NE tagger we use is a derivative of that described in (Prager et al., to appear). It tags named entities of about 100 types. Some of the marked types are very specific, like ZIPCODE and ROYALTY. To avoid overfitting we choose six high-level types for atomic events' extraction: PERSON, PLACE, DATE, WHOLENO, ORG and ROLE. In contrast to the original atomic event scores we keep simple counts for the triplets as later we combine triplets extracted for different people. Table 1 contains examples from the list of atomic events extracted for Columbus.

3.3 Generalized atomic events

Our goal is to collect information about activities general for all people and about activities specific for some occupations. Thus, we are interested in the semantic information reflected in the atomic events but not in the exact named entities. We analyze not the atomic events themselves but the generalized versions of the extracted atomic events. For example, here are two sentences about explorers:

> Vespucci explored the shores of South America.
> Vitus Bering explored Aleutian Islands.

The corresponding atomic events extracted for these sentences are presented in Table 2. Clearly, these atomic events capture information about the same type of activity, namely that explorers explore some locations. What makes these atomic events different is the exact names of the explorers and the locations a particular explorer explored. We can unify these atomic events by omitting the exact named entities and leaving only their types. The resulting atomic events we call *generalized atomic events*. The atomic events presented in Table 2 can be converged to the following generalized atomic event:

NAME/PERSON - explored/VBD - PLACE

In the generalized atomic events we distinguish two types of named entities with the tag PERSON: those which refer to the person under analysis (from now on they are marked as NAME/PERSON) and all the rest (marked as PERSON). Thus, we separate the person whose occupation we want to identify from the people who are linked to this person through some activities. This generalization technique is similar to the one used by Yangarber(2003) for semantic patterns discovery for information extraction.

Filatova and Hatzivassiloglou (2003) showed that atomic events capture the most important relations described in the input and assign to them good-quality labels. In this work we show that generalized atomic events can be used for capturing the activities performed by people of different occupations.

To select occupation-related activities we merge lists of atomic events corresponding to the people of the same occupation. Hence, we get ten lists of generalized atomic events corresponding to the ten occupations under analysis. The count of each generalized atomic event is equal to the sum of the counts of all the atomic events which are merged into this generalized atomic event.

4 Getting occupation-related activities

We assume that the activities important for an occupation are linked to the named entities important for this occupation and vice versa, the named entities important for this occupation are linked to the representatives of this occupation through the important activities. Formulated like this, the problem of identifying the actions important for an occupation can be solved using the methodology suggested (pre-Google) by Kleinberg (1998) for ranking web-sites, where a search engine counts "inbound and outbound links to identify central sites in a community." The major idea of this technique is based on the assumption that good hubs contain links to good authorities and that links to good authorities are listed within good hubs. Treating activity verbs as hubs and named entity tags as authorities we map the problem of discovering activities closely related to a specific occupation to the problem of ranking the reliability of web-pages for the submitted query.

To rank the importance of activities for a particular occupation we define a bipartite graph $G = \{N, V, E\}$, where V are the verb nodes (activities), N are the nodes corresponding to the named entity types linked to V verbs, and E are the arcs connecting the named entity types and the activities. A part of such a bipartite graph created for the *explorers* occupation is presented in Figure 1.

Following this procedure we create bipartite graphs for every occupation. Each of the created ten bipartite graphs contains m named entity types on one side and k verbs (activities) on the other side.[3]

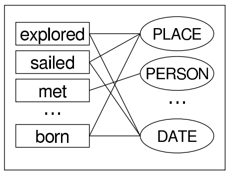


Figure 1: Bipartite graph for a set of generalized atomic events corresponding to *explorers* occupation

Dancer	Physicist	Singer
made/VBD	born/VBN	said/VBD
died/VBD	died/VBD	born/VBN
appeared/VBD	announced/VBD	died/VBD
been/VBN	discovered/VBD	join/VB
founded/VBD	be/VB	singing/VBG
became/VBD	including/VBG	sang/VBD
born/VBN	became/VBD	has/VBZ
danced/VBD	wrote/VBD	conducting/VBG
blessed/VBN	helped/VBD	made/VBD
perform/VB	named/VBN	became/VBD

Table 3: Top ten activities for four occupations: dancers, physicists and singers

We define $P_{N \to V}$ as a $m \times k$ stochastic transition matrix from named entities to verbs, with elements[4]

$$P_{N \to V}[i,j] \equiv p_{n_i,v_j} = (1-c)\frac{f(n_i \to v_j)}{\sum_{v \in V} f(n_i \to v)} + c \quad (1)$$

where f is equal to the sum of the counts of all the generalized atomic events containing this link for the occupation under analysis. In the same way we define $P_{V \to N}$ $k \times m$ row-stochastic transition matrix from verbs to named entities. Using $P_{V \to N}$ and $P_{N \to V}$, we can define the transition matrix:

$$P_{V \to V} = P_{V \to N} \cdot P_{N \to V} \quad (2)$$

that can be used for scoring the verbs according to how important they are for the current occupation. Due to the construction rules, this matrix is stochastic.According to Markov Chain Theory (Kemeny and Snell, 1960) for a square stochastic matrix it is possible to find a steady state which corresponds to the eigenvector for the eigenvalue equal to 1. Any square stochastic matrix has 1 among its eigenvalues. The same way the eigenvector corresponding to the steady state for web-pages ranks these pages, the eigenvector corresponding to the steady state of transition matrix (2) ranks how tightly the activities

[3]Variables m and k are unique for every occupation

[4]To avoid data sparseness we use a smoothing factor $c = 0.01$.

Biography-related verbs
said/VBD
born/VBN
died/VBD
wrote/VBD
became/VBD
had/VBD
known/VBN
be/VB
included/VBD
including/VBG

Table 4: Top ten activities common for all the eleven occupations.

are linked to the occupation under consideration. The size of this matrix depends on the variety of the verbs in all forms used in the generalized atomic events for the representatives of this occupation and varies from 800 for physicists up to 2100 for politicians in our data.

Table 3 contains top ten activities for three occupations: dancers, physicists and singers. These activities are listed in the sorted order, the ones on top of the table have the highest scores in the respective eigenvectors corresponding to the eigenvalues of 1.

The activities presented in Table 3 can be divided into three types:

1. those which are occupation-specific, such as *danced* and *perform* for dancers, *discovered* for physicists, *singing* and *sang* for singers;
2. those which are likely to be used in any biography, such as *born*, *died*, *became*;
3. other, which are mostly general purpose verbs, such as *been*, *made*.

For our classification we rely mainly on the first type of the activities. To extract the activities of the second type we create $P_{V \rightarrow V}$ a transition matrix for the combined set for all the generalized atomic events created for all the ten occupations. This matrix is also stochastic and by calculating the eigenvector corresponding to its steady state we can identify those activities which are tightly linked to any person irrespectively of his/her occupation and thus reflect general biographical information. Table 4 contains top ten activities for this matrix.

In Section 5 we show that the lists of occupation-related and general activities are reliable features for classifying people according to their occupations.

5 Classification

In this section we describe people classification according to their occupations. For our classification experiments we use a multi-class SVM classifier.[5] As we have 189 data-points corresponding to ten classes we use leave-one-out cross validation which allows us to use the maximal possible amount of data for training. We experiment with two sets of features: one set consists only of the verbs corresponding to the occupation-specific activities (Section 5.1); the other set consists of the complete triplets for the generalized atomic events (Section 5.2).

5.1 SVM classification using only verbs

To get verb features for multi-class SVM classification we use ten occupation-related lists of activities with activities sorted according to the eigenvector corresponding to the steady state.

The verb-only algorithm is as follows:

V1 Get the sorted list of activities (verbs) for every occupation (ten lists). These activities are the major features on which SVM relies to assign an occupation to a person.

V2 Get the sorted list of activities for all occupations merged together. These activities are used in Step V4 to remove from the list of classification features those activities which are general and not helpful for identifying the occupation of a person.

V3 Get top 15% of the activities from each of the ten lists.

V4 From the ten occupation-specific lists remove those activities which are also present in the list of general activities.

V5 Merge ten occupation-related lists into one list and remove from this list all the activities that appear in more than 2 occupations.

By leaving at Step 3 some percentage of the activities (verbs) instead of an absolute amount, we take into account the fact that the number of activities used to describe different occupations varies from occupation to occupation (for example, 1794 activities are used in the atomic events for composers, and 800 - for physicists). As the activities get scores according to the steady state vectors, the activities with high scores are the ones which are most likely to be used for the description of a person of the current occupation. The activities with low values are too specific and are likely to be used in only a few descriptions of people of this occupation. For example, we know that Alexander Borodin was both a composer and a chemist: we do not want to keep those specific verbs which describe his activities as a chemist in the list of the activities describing composers.

In Step 4 we remove from our final list those activities that are typical for all humans and thus cannot

[5] `http://www.cs.cornell.edu/People/tj/svm_light/svm_multiclass.html`

			SVM classification				Probing		Random
			Verbs		Atomic Events				
Occupation	Amount of Representatives	Average Amount of Documents	Amount	Ratio	Amount	Ratio	Amount	Ratio	
Artists	20	10.0	9	0.450	15	0.750	14	0.700	0.106
Athletes	18	10.0	12	0.667	16	0.889	14	0.778	0.095
Composers	20	9.65	10	0.500	15	0.750	19	0.950	0.106
Dancers	15	9.07	7	0.467	13	0.867	11	0.733	0.079
Explorers	17	9.0	12	0.706	15	0.882	15	0.882	0.090
Mathematicians	20	7.2	10	0.500	8	0.381	20	1.000	0.106
Physicists	19	7.05	5	0.263	6	0.316	13	0.684	0.101
Politicians	20	10.0	9	0.450	12	0.600	1	0.050	0.106
Singers	20	9.05	9	0.450	12	0.600	10	0.500	0.106
Writers	20	10.0	7	0.350	12	0.600	10	0.500	0.106
Average				0.480		0.663		0.677	

Table 5: Performance of different classification methods.

be used to distinguish among different occupations. In Step 5 we make our activities as specific as possible: For example, there will be some intersection in activities among mathematicians and physicists, and such activities cannot be helpful for differentiation between these occupations.

The final activities list is used as the list of features for SVM classification. Then we assign values to these features for every person: if the activity from the features list is used as a connector for the extracted atomic events, then this feature receives the value of 1, if there is no atomic event using this activity as a connector then this feature is assigned 0. We use binary values for our features instead of the atomic event counts because the reliability of the scores for the atomic events extracted for different people varies greatly. For some people we retrieve 10 documents with many biographical facts about those people, for other people we retrieve 2 or 3documents which only mentione the people queried.

Removal of some of the features is reinforced by the (Koller and Sahami, 1997) work on feature selection for document classification. It indicates that keeping only a small fraction of the available features improves the classification performance. The optimal number of features is still to be determined.

We train our classifier and evaluate its performance using leave-one-out cross validation. Out of 189 people, eight are not assigned any features. This is because all the atomic events extracted for these 8 people are either too general or too specific. As these 8 people do not have any features to assign them to the most likely occupation, they are misclassified to the default occupation (artists). Six of these eight are mathematicians, one is a dancer, one is a physicist. Absence of verbal features can be explained by a small number of the documents retrieved for these people. Table 5 shows how many documents are analyzed per person on average for each occupation. Due to the nature of our document collections, the smallest number of documents analyzed was for mathematicians and physicists: current newswire texts do not contain much information about scientists, and those parts of encyclopedias which we had at our disposal only contained information for some of the scientists. Out of the remaining 181 people, only 90 are classified correctly. As the distribution of people across occupations is not even, Table 5 contains two numbers for each occupation: the absolute number and the ratio of people classified correctly for this occupation.

We believe that the performance of SVM classification based solely on the activities is so poor because it does not take into account the information that many activities which are expressed with the help of the same verb are surrounded by different types of arguments for different occupations. For example, Henri Matisse is classified as a dancer based on the frequent co-occurrence with the *dance* activity, which is understandable as one of his most famous paintings is "Dance". Or, the *explored* activity is among the top activities for several occupations: writers, composers, explorers; but only for the explorers this activity is linked to the PLACE named entity tag. Though the classification based solely on verbs gives quite poor results we consider it to be a valid starting classification as usually activities are associated with the verbs corresponding to these activities.

5.2 SVM classification using atomic events

To create generalized atomic event features for multiclass SVM classification we use the sorted lists of activities for the ten occupations and the general list of activities. The activities are sorted according to the values they get from the eigenvector corresponding to the steady state.

AE1 Same as step V1 above.
AE2 Same as step V2 above.

AE3 For the top 15% of the activities from the ten occupation-specific lists get all the generalized atomic events containing those activities.

AE4 For the top 15% of the activities typical for all the occupation (Step AE2) get all the generalized atomic events containing those activities.

AE5 From the ten occupation-related lists (Step AE3) remove those generalized atomic events which are also present in the list of general generalized atomic events (Step AE4).

AE6 Merge the ten occupation-related lists into one and remove from it all the generalized atomic events that appear in more than 2 occupations.

Out of 189 people, nine are not assigned any features. This again is because all the atomic events extracted for these 9 people were either too general or too specific. The people who do not get any event features are the same as those who do not get any verb features plus one physicist. Out of the remaining 180 people, 124 people are classified into the appropriate occupations. Table 5 shows that generalized atomic events are more reliable for occupation classification than plain activities extracted from these generalized atomic events. Thus, it can be concluded that structured information captured by atomic events is valuable and reliable. For example, using atomic events Matisse was correctly classified as an artist. According to the t-test the performance of the classification based on atomic events is significantly better ($p < 0.05$) than the performance of the classification based solely on activities.

We would like to note, that after closer analysis some of the cases of misclassification can be considered as correct assignments as a person could excel in different occupations. For example, in our corpus Paul McCartney is defined as a singer while classifying him as a composer is a valid results as well.

5.3 Other types of classification

The task of classifying people according to their occupations is new and to our knowledge there is no existing baseline we could compare our results with. Nevertheless, we decided to adapt for comparison two classification techniques used for other tasks: random assignment of an occupation and probing.

Random occupation assignment. As the distribution of people among the occupations is not even we cannot give one exact probability of assigning a correct occupation to a particular person. Instead, we calculate such random probabilities for each occupation. The results are presented in Table 5.

Random assignment gives a very low baseline which we easily outperform, which is why we use another classification based on probing to estimate how good our results are. Classification based on probing is considered to be the state-of-the-art classification method for such tasks as hidden web classification (Ipeirotis et al., 2003) and answer verification (Magnini et al., 2002).

Probing. First, we get the counts of how many documents are retrieved for the queries containing only the titles of the occupations (e.g., "⟨*role*⟩ *mathematician* ⟨*/role*⟩", "⟨*role*⟩ *artist* ⟨*/role*⟩", etc.). Then, we get the counts for the queries containing all possible combinations of people's names and occupations' titles. (for example, "⟨*role*⟩ *mathematician* ⟨*/role*⟩ ⟨*person*⟩ *Picasso*⟨*/person*⟩", "⟨*role*⟩ *artist* ⟨*/role*⟩ ⟨*person*⟩ *Picasso* ⟨*/person*⟩", etc.). Finally, we divide the counts for the queries submitted for the occupation plus person by the count for the corresponding occupation query. The maximum of all the ratios for the person gives the occupation for this person.

$$Occupation_j = max_{for\ all\ i,j} \frac{count_{occupation_i,\ person_j}}{count_{occupation_i}} \quad (3)$$

According to Table 5 SVM, classification based on atomic events outperforms probing classification for six occupations out of ten, for one occupation (*explorers*) the results for SVM classification and probing are the same and for three occupations probing classification outperforms SVM classification. One of the cases where probing classification outperforms SVM classification is *mathematicians*, where nine mathematicians have no features in SVM classification and thus, do not have any better than random chance to be classified correctly.

Thus, our SVM classification of people according to their occupations based on atomic events has performance comparable to probing-based classification. This is significant since in those tasks for which it has been used so far, probing classification outperforms other methods and is considered to be the state-of-the-art (Ipeirotis et al., 2003; Magnini et al., 2002).

5.4 Using classification extracted features

Though we do not dramatically outperform probing, our methodology has one crucial advantage. We use classification not as a primary task but as an evaluation testbed to show that the lists of generalized atomic events created for every occupation and for general biographies indeed capture the major activities performed by people of the respective occupa-

Artists	Athletes
NAME - painted/VBN - DATE	NAME - win/VB - WHOLENO
NAME - resemble/VB - PERSON	NAME - scored/VBD - WHOLENO
PERSON - designed/VBN - NAME	NAME - winning/VBG - DATE
Composers	Dancers
NAME - composed/VBN - PERSON	NAME - danced/VBN - ORG
NAME - include/VBP - WHOLENO	PLACE - presented/VBD - NAME
ROLE - hearing/VBG - NAME	NAME - appeared/VBD - WHOLENO
Explorers	Mathematicians
NAME - annexes/VBZ - PLACE	PERSON - developed/VBD - NAME
NAME - reach/VB - PLACE	ROLE - prove/VB - NAME
NAME - declares/VBZ - DATE	NAME - studied/VBD - WHOLENO
Physicists	Politicians
DATE - described/VBD - NAME	NAME assassinated/VBN - PLACE
ROLE - predicted/VBD - NAME	NAME - postponed/VBN - PLACE
NAME - continued/VBD - ORG	NAME - flown/VBN - ORG
Singers	Writers
NAME - conducting/VBG - PLACE	PLACE - leaving/VBG - NAME
NAME - sing/VB - ROLE	NAME - translated/VBN - PERSON
NAME - sang/VBD - PERSON	WHOLENO - written/VBN - NAME

Table 6: Occupation-specific generalized atomic events (NAME stands for NAME/PERSON).

tions and can be used for biography generation. For example, the generalized atomic events which are used for the description of representatives within all the ten occupations and are excluded from the list of features for SVM classification as too general, contain verbs such as *born/VBN*, *died/VBD* linked to the DATE and PLACE named entity tags or *became/VBD* linked to the ROLE named entity tag. Table 6, on the other hand, contains occupation-specific generalized atomic events. These generalized atomic events have high scores within the respective occupations, are used as features for SVM classification and have non-zero values in the feature sets which correctly classified people into the appropriate occupations.

6 Conclusions and future work

We reported results on extracting human activities which can be used for classifying people according to their occupations. We introduced a novel representation for describing human activities (generalized atomic events). SVM classification using generalized atomic events as features gives results comparable to other state-of-the-art classification techniques. We are currently looking at ways of identifying other types of activities which are neither general no occupation-specific but rather person-specific. We observed that those generalized atomic events which have high scores for a particular person but are not used in the description of any other person are good candidates to point out person-specific information. We believe that the usage of generalized atomic events can enable significant new techniques for a number of natural language processing tasks. One direction is to use the generalized atomic events typical for all the occupations as an initial representation for the auxiliary biography-related questions. Another direction is to use generalized atomic events for biography generation.

References

M. Biryukov, R. Angheluta, and M.-F. Moens. 2005. Multi-document question answering text summarization using topic signatures. *Journal on Digital Information Management*, 3(1):27–33.

S. Blair-Goldensohn, D. Evans, V. Hatzivassiloglou, K. McKeown, A. Nenkova, R. Passonneau, B. Schiffman, A. Schlaikjer, A. Siddharthan, and S. Siegelman. 2004a. Columbia university at DUC 2004. In *Proceedings of DUC*, Boston.

S. Blair-Goldensohn, K. McKeown, and A. Schlaikjer, 2004b. *Answering Definitional Questions: A Hybrid Approach*, pages 47–58. AAAI Press.

D. Carmel, E. Amitay, M. Hersovici, Y. Maarek, Y. Petruschka, and A. Soffer. 2001. Juru at TREC 10. Experiments with index pruning. In *Proceedings of TREC*.

P. Duboue and K. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the EMNLP Conference*, pages 121–128, Sapporo, Japan.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT press.

E. Filatova and V. Hatzivassiloglou. 2003. Domain-independent detection, extraction, and labeling of atomic events. In *Proceedings of the RANLP Conference*, pages 145–152, Bulgaria.

P. Ipeirotis, L. Gravano, and M. Sahami. 2003. QProber: A system for automatic classification of hidden-web resources. *ACM Transactions on Information Systems*, 21(1):1–41.

J. Kemeny and J. Snell. 1960. *Finite Markov Chains*. Princeton, NJ: Van Nostrand.

J. Kleinberg. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677.

D. Koller and M. Sahami. 1997. Hierarchically classifying documents using very few words. In *Proceedings of ICML*, pages 170–178, Nashville, US.

C.-Y. Lin and E. Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the COLING Conference*, Germany, July.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2002. Is it the right answer? Exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual ACL Meeting*, pages 425–432, Philadelphia, USA.

J. Prager, J. Chu-Carroll, and K. Czuba. 2004. Question answering using constraint satisfaction: QA-by-Dossier-with--Constraints. In *Proceedings of the 42nd Annual ACL Meeting*, pages 575–582, Spain.

J. Prager, J. Chu-Carroll, E. Brown, and K. Czuba, to appear. *Question Answering by Predictive Annotation*. Kluwer Academic Publishers.

B. Schiffman, I. Mani, and K. Concepcion. 2001. Producing biographical summaries: combining linguistic knowledge with corpus statistics. In *Proceedings of the 39th Annual ACL Meeting*, pages 450–457, Toulouse, France.

R. Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the ACL Conference*, Japan.

L. Zhou, M. Ticrea, and E. Hovy. 2004. Multi-document biography summarization. In *Proceedings of the EMNLP Conference*, pages 434–441, Spain.

Using Names and Topics for New Event Detection

Giridhar Kumaran and James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
{giridhar,allan}@cs.umass.edu

Abstract

New Event Detection (NED) involves monitoring chronologically-ordered news streams to automatically detect the stories that report on new events. We compare two stories by finding three cosine similarities based on names, topics and the full text. These additional comparisons suggest treating the NED problem as a binary classification problem with the comparison scores serving as features. The classifier models we learned show statistically significant improvement over the baseline vector space model system on all the collections we tested, including the latest TDT5 collection.

The presence of automatic speech recognizer (ASR) output of broadcast news in news streams can reduce performance and render our named entity recognition based approaches ineffective. We provide a solution to this problem achieving statistically significant improvements.

1 Introduction

The instant and automatic detection of new events is very useful in situations where novel information needs to be detected from a real-time stream of rapidly growing data. These real-life situations occur in scenarios like financial markets, news analyses, and intelligence gathering. In this paper we focus on creating a system to immediately identify stories reporting new events in a stream of news - a daunting task for a human analyst given the enormous volume of data coming in from various sources.

The Topic Detection and Tracking (TDT) program, a DARPA funded initiative, seeks to develop technologies that search, organize and structure multilingual news-oriented textual materials from a variety of broadcast news media. One of the tasks in this program, New Event Detection (NED), involves constant monitoring of streams of news stories to identify the first story reporting topics of interest. A *topic* is defined as "a seminal event or activity, along with directly related events and activities" (Allan, 2002). An earthquake at a particular place is an example of a topic. The first story on this topic is the story that first carries the report on the earthquake's occurrence. The other stories that make up the topic are those discussing the death toll, the rescue efforts, the reactions from different parts of the world, scientific discussions, the commercial impact and so on. A good NED system would be one that correctly identifies the article that reports the earthquake's occurrence as the first story.

NED is a hard problem. For example, to distinguish stories about earthquakes in two different places, a vector space model system would rely on a tf-idf weighting scheme that will bring out the difference by weighting the locations higher. More often then not, this doesn't happen as the differences are buried in the mass of terms in common between stories describing earthquakes and their aftermath. In this paper we reduce the dependence on tf-idf weighting by showing the utility of creating three distinct representations of each story based on named entities. This allows us to view NED as a binary classification problem - i.e., each story has to be classified into one of two categories - *old* or *new*, based on features extracted using the three different representations.

The paper starts by summarizing the previous work on NED in Section 2. In Section 3, we explain the rationale behind our intuition. Section 4 describes the experimental setup, data pre-processing, and our baseline NED system. We then briefly describe the evaluation methodology for NED in Section 5. Model creation and the results of applying these models to test data are detailed in Section 6. In the same section, we describe the effect on performance if the manually transcribed version of broadcast news is replaced with ASR output. Since its hard to recognize named entities from ASR data, performance expectedly deteriorates. We follow a novel approach to work around the problem resulting in statistically significant improvement in performance. The results are analyzed in Section 7. We wrap up with conclusions and future work in Section 8.

2 Previous Research

Previous approaches to NED have concentrated on developing similarity metrics or better document representations or both. A summer workshop on topic-based novelty detection held at Johns Hopkins University extensively studied the NED problem. Similarity metrics, effect of named entities, pre-processing of data, and language and Hidden Markov Models were explored (Allan et al., 1999). Combinations of NED systems were also discussed. In the context of this paper, selective re-weighting of named entities didn't bring about expected improvement.

Improving NED by better comparison of stories was the focus of following papers. In an approach to solve on-line NED, when a new document was encountered it was processed immediately to extract features and build up a query representation of the document's content (Papka and Allan, 1998). The document's initial threshold was determined by evaluating it with the query. If the document did not trigger any previous query by exceeding this particular threshold, it was marked as a new event. Unlike the previous paper, good improvements on TDT benchmarks were shown by extending a basic incremental TF-IDF model to include source-specific models, similarity score normalization techniques, and segmentation of documents (Brants et al., 2003).

Other researchers have attempted to build better document models. A combination of evidence derived from two distinct representations of a document's content was used to create a new representation for each story (Stokes and Carthy, 2001). While one of the representations was the usual free text vector, the other made use of lexical chains (created using WordNet) to obtain the most prevalent topics discussed in the document. The two vectors were combined in a linear fashion and a marginal increase in effectiveness was observed.

NED approaches that rely on exploiting existing news tracking technology were proved to inevitably exhibit poor performance (Allan et al., 2000). Given tracking error rates, the lower and upper bounds on NED error rates were derived mathematically. These values were found to be good approximations of the true NED system error rates. Since tracking and filtering using full-text similarity comparison approaches were not likely to make the sort of improvements that are necessary for high-quality NED results, the paper concluded that an alternate approach to NED was required. This led to a series of research efforts that concentrated on building multi-stage NED algorithms and new ways to combine evidence from different sources.

In the topic-conditioned novelty detection approach, documents were classified into broad topics and NED was performed within these categories (Yang et al., 2002). Additionally, named entities were re-weighted relative to the normal words for each topic, and a stop list was created for each topic. The experiments were done on a corpus different from the TDT corpus and, apparently didn't scale well to the TDT setting.

The DOREMI research group treated named entities like *people* and *locations* preferentially and developed a new similarity measure that utilized the semantics classes they came up with (Makkonen et al., 2002). They explored various definitions of the NED task and tested their system accordingly. More recently, they utilized a perceptron to learn a weight function on the similarities between different semantic classes to obtain a final confidence score for each story (Makkonen et al., 2004).

The TDT group at UMass introduced multiple document models for each news story and modified similarity metrics by splitting up stories into only named entities and only terms other than named entities (Kumaran and Allan, 2004). They observed that certain categories of news were better tackled using only named entities, while using only topic terms for the others helped.

In approaches similar to named entity tagging, part-of-speech tagging (Farahat et al., 2003) has also been successfully used to improve NED.

Papers in the TDT2003 and TDT2004 workshops validated the hypothesis that ensemble single-feature classifiers based on majority voting exhibited better performance than single classifiers working with a number of features on the NED task (Braun and Kaneshiro, 2003; Braun and Kaneshiro, 2004). Examples of features they used are cosine similarity, text tiling output and temporally-weighted tf-idf.

Probabilistic models for online clustering of documents, with a mechanism for handling creation of new clusters have been developed. Each cluster was assumed to correspond to a topic. Experimental results did not show any improvement over baseline systems (Zhang et al., 2005).

3 Features for NED

Pinning down the character of new stories is a tough process. New events don't follow any periodic cycle, can occur at any instant, can involve only one particular type of named entity (people, places, organizations etc.) or a combination, can be reported in any language, and can be reported as a story of any length by any source[1]. Apart from the source, date, and time of publication or broadcast of each news story, the TDT corpora do not contain any other clues like placement in the webpage, the number of sources reporting the same news and so on. Given all these factors, we decided that the best features to use would be those that were not particular to the story in question only, but those that measure differences between the story and those it is compared with.

[1]It could be argued that articles from a source, say *NYTimes*, are much longer than news stories from *CNN*, and hence the length of stories is a good candidate for use as a feature. However, when there is no pattern that indicates that either of the two sources reports new stories preferentially, the use of length as a feature is moot.

Category-specific rules that modified the baseline confidence score assigned to each story have been developed (Kumaran and Allan, 2004). The modification was based on additional evidence in the form of overlap of named entities and topic terms (terms in the document not identified as named entities) with the closest story reported by a baseline system. We decided to use these three scores: namely the baseline confidence score, named entity overlap, and topic-term overlap as features. The named entities considered were *Event*, *Geopolitical Entity*, *Language*, *Location*, *Nationality*, *Organization*, *Person*, *Cardinal*, *Ordinal*, *Date*, and *Time*. These named entities were detected in stories using BBN IdentiFinder^{TM}(Bikel et al., 1999). Irrespective of their type, all named entities were pooled together to form a single named entity vector.

The intuition behind using these features is that we believe every event is characterized by a set of people, places, organizations, etc. (named entities), and a set of terms that describe the event. While the former can be described as the *who*, *where*, and *when* aspects of an event, the latter relates to the *what* aspect. If two stories were on the same topic, they would share both named entities as well as topic terms. If they were on different, but similar, topics, then either named entities or topic terms will match but not both.

We illustrate the above intuition with examples. Terms in **bold face** are named entities common to both stories, while those in *italics* are topic terms in common. We start with an example showing that for old stories both the named entities as well as topic terms overlap with a story on the same topic.

Story 1. : Story on a topic already reported

While in **Croatia** today, **Pope John Paul II** called on the *international community* to *help* end the fighting in the Yugoslavia's **Kosovo** province.

Story 2. : Story on the same topic

Pope John Paul II is urging the *international community* to quickly *help* the ethnic Albanians in **Kosovo**. He spoke in the coastal city of Split, where he ended a three-day visit to **Croatia**.

Story 1 is an old story about Pope John Paul II's visit to Yugoslavia. Story 2 was the first story on the topic and it shares both named entities likes **Pope John Paul II** and **Croatia** and also topic terms like *international community* and *help*.

Our next example shows that for new stories, either the named entities or topic terms match with an earlier story.

Story 3. : Topic not seen before

Turkey has sent 10,000 troops to its southern border with Syria amid growing tensions between the two neighbors, newspapers reported Thursday. Defense Minister **Ismet Sezgin** denied any troop movement along the border, but said **Turkey's** patience was running out. **Turkey** accuses Syria of harboring Turkish Kurdish rebels fighting for autonomy in **Turkey's** southeast; it says rebel leader Abdullah Ocalan lives in Damascus.

Story 4. : Closest Story due to Named Entities

A senior **Turkish** government official called Monday for closer military cooperation with neighboring Bulgaria. After talks with President Petar Stoyanov at the end of his four-day visit, Turkish Deputy Premier and National Defense Minister **Ismet Sezgin** expressed satisfaction with the progress of bilateral relations and the hope that Bulgarian-**Turkish** military cooperation will be promoted.

Story 3 is a new story about the rising tensions between Turkey and Syria. The closest story as reported by a (baseline) basic vector space model NED system using cosine similarity is Story 4, a story about Turkish-Bulgarian relations. The named entities **Turkey** and **Ismet Sezgin** caused this match. We see that none of the topic terms match. However, the system reported with a high confidence score that Story 3 is old. This is because of the matching of high IDF-valued named entities. Determining that the topic terms didn't match would have helped the system avoid this mistake.

4 Experimental Setup and Baseline

We used the TDT2, TDT3, TDT4, and TDT5 corpora for our experiments. They contain a mix of broadcast news (*bn*) and newswire (*nwt*) stories. Only the English stories in the multi-lingual collections were considered for the NED task. The broadcast news material is provided in the form of an audio sampled data signal, a manual transcription of the audio signal (*bn-man*), and a transcription created using an automatic speech recognizer (*bn-asr*).

We used version 3.0 of the open source Lemur system[2] to tokenize the data, remove stop words, stem and create document vectors. We used the 418 stopwords included in the stop list used by InQuery (Callan et al., 1992), and the Krovetz-stemmer algorithm implementation provided as part of Lemur.

Documents were represented as term vectors with incremental TF-IDF weighting (Brants et al., 2003; Yang et al., 1998). We used the cosine similarity metric to judge the similarity of a story *S* with those seen in the past.

$$Sim(S,X) = \frac{\sum_w weight(w,S) * weight(w,X)}{\sqrt{\sum_w weight(w,S)^2}\sqrt{\sum_w weight(w,X)^2}} \quad (1)$$

where

$$\begin{aligned} weight(w,d) &= tf * idf \\ tf &= \log(termfrequency + 1.0) \\ idf &= \frac{\log((docCount+1)}{(documentfreq+0.5)} \end{aligned}$$

The maximum similarity of the story *S* with stories seen in the past was taken as the confidence score that *S* was old. This constituted our baseline system.

We extracted three features for each incoming story *S*. The first was the confidence score reported by the baseline system. The second and third features were the cosine similarity between only the named entities in *S* and *X* and the cosine similarity between only the topic terms in *S* and *X*. We trained a Support Vector Machine (SVM) (Burges, 1998) classifier on these features. We chose to use SVMs as they are considered state-of-the-art for text classification purposes (Mladeni et al., 2004), and provide us with options to consider both linear and nonlinear decision boundaries. To develop SVM models we used SVM^{Light}(Joachims, 1999), which is an implementation of SVMs in C. SVM^{Light} is an implementation of Vapnik's Support Vector Machine (Vapnik, 1995).

For training, we used the TDT3 and TDT4 corpora. There were 115 and 70 topics respectively giving us a total of 185 positive examples (new stories)

[2] `http://www.lemurproject.org`

and 7800 negative examples (old stories). We balanced the number of positive and negative examples by oversampling the minority class until there were equal number of positive and negative training instances. Testing was done on the TDT2 and TDT5 corpora (96 and 126 topics resp.).

5 NED Evaluation

The official TDT evaluation requires a NED system to assign a confidence score between 0 (new) and 1 (old) to every story upon its arrival in the time-ordered news stream. This assignment of scores is done either immediately upon arrival or after a fixed look-ahead window of stories. To evaluate performance, the stories are sorted according to their scores, and a threshold sweep is performed. All stories with scores above the threshold are declared *old*, while those below it are considered *new*. At each threshold value, the misses and false alarms are identified, and a cost C_{det} is calculated as follows.

$$C_{det} = C_{miss} * P_{miss} * P_{target} + C_{FA} * P_{FA} * P_{non-target}$$

where C_{Miss} and C_{FA} are the costs of a Miss and a False Alarm, respectively, P_{Miss} and P_{FA} are the conditional probabilities of a Miss and a False Alarm, respectively, and P_{target} and $P_{non-target}$ are the a priori target probabilities ($P_{non-target}$ = 1 - P_{target}).

The threshold that results in the least cost is selected as the optimum one. Different NED systems are compared based on their minimum cost. In other words, the **lower** the C_{det} score reported by a system on test data, the **better** the system.

6 Results

Our first set of experiments were performed on data consisting of newswire text and manual transcription of broadcast news (*nwt+bn-man*). We used the features mentioned in Section 3 to build SVM models in the classification mode. We experimented with linear, polynomial, and RBF kernels. The output from the SVM classifiers was normalized to fall within the range zero and one.

We found that using certain kernels improved performance over the baseline system significantly. The results for both corpora, TDT2 and TDT5, were consistently and significantly improved by using the classification models. The 2004 NED evaluations conducted by the National Institute of Standards and Technology was on the TDT5 collection. The large size of the collection and existence of a large number of topics with a single story made the task very challenging. The best system fielded by the participating sites was the baseline system used here. Table 1 summarizes the results we obtained.

Kernel Type	TDT2 (*nwt+bn-man*)	TDT5 (*nwt*)
Baseline System	0.585	0.701
Linear Kernel	0.548	0.696
Poly. of deg. 1	0.548	0.696
Poly. of deg. 2	0.543	0.688
Poly. of deg. 3	0.545	0.684
Poly. of deg. 4	0.535	0.694
Poly. of deg. 5	0.533	0.688
Poly. of deg. 6	0.534	0.693
RBF with γ = 1	**0.540**	**0.661**
RBF with γ = 5	0.530	0.699

Table 1: Summary of the results of using SVM classifi er models for NED on the TDT2 and TDT5 collections. The numbers are the minimum cost (C_{det}) values (lower is better). The sign test, with α = 0.05, was performed to compare the baseline system with only a classifi er using RBF kernels with γ = 1. For both collections, the improvements were found to be statistically signifi cant (shown in bold). While there are better performing kernels for TDT2, we chose to perform signifi cance tests for only one kernel to show that signifi cant improvement over the baseline can be obtained using a single kernel across different test collections.

All statistical significance testing was done using the sign test. We counted the number of topics for which using the SVM classifier improved over the baseline (in terms of detecting more previously undetected new and old stories), and also the number of topics for which using the SVM classifier actually converted originally correct decisions into wrong ones. These were used as input for the sign test. The test were used to check whether improvement in performance using the classifier-based system was spread across a significant number of topics, and not confined to a few. Table 2 gives some examples of topics and the associated improvements in detecting them.

Topic ID	Number of old stories	Num. detected by baseline system	Num. detected by SVM classifier	Improvement (Higher the better)
55105	420	407	403	-4
55010	21	21	20	-1
55023	5	5	4	-1
55089	226	226	225	-1
55125	120	114	120	6
55107	331	327	331	7
55106	808	787	795	8
55200	196	185	193	8

Table 2: Examples of improvements due to using the SVM classifier on a per-topic basis. Shown here are the four topics each in which the greatest degradation and improvements in performance were seen. The topics vary in size. The SVM classifier resulted in overall (statistically significant, refer Table 1) improvement as it corrected more errors than introduced them.

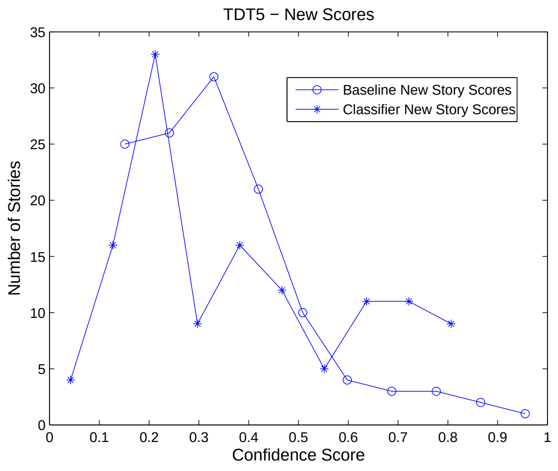


Figure 1: Distribution of new story scores for the baseline and SVM model systems.

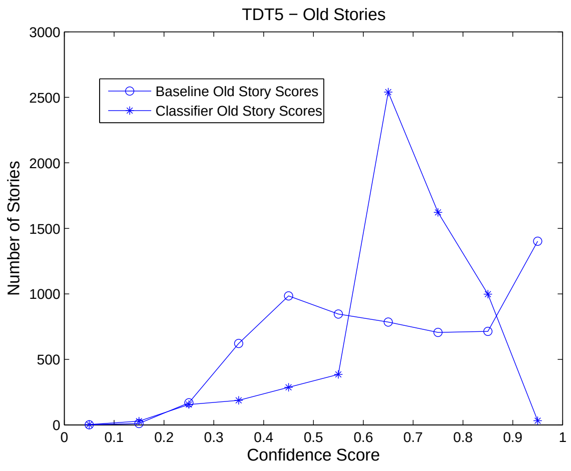


Figure 2: Distribution of old story scores for the baseline and SVM model systems.

7 Analysis

The main goal of our effort was to come up with a way to correctly identify new stories based on features we thought characterized them. To understand what we had actually achieved by using these models, we studied the distribution of the confidence scores assigned to new and old stories for the baseline and a classifier-based NED system for the TDT5 collection (Figures 1 and 2 respectively).

We observe that the scores for a small fraction of new stories that were initially missed (between scores 0.8 and 1) are decreased by the model-based NED system while a larger fraction (between scores 0.1 and 0.4) is also decreased by a small amount. However, the major impact of using SVM model-based NED systems appears to be in detecting old stories. We observe that the scores of a significant number of old stories (between scores 0.2 and 0.55) have been increased to be closer to one. This had the effect of increasing the score difference between old and new stories, and hence improved classification accuracy as measured by the minimum cost.

We investigated the relative importance of the three features by looking that the linear kernel SVM model. While the original cosine similarity metric *CS* remained the prominent feature, the contribution of the third feature *non-NE-CS* was slightly more than if not equal to the contribution of named entities *NE-CS* (Table 3). This explains why simple re-weighting of named entities alone (Allan et al., 1999) doesn't suffice to improved performance.

Feature	*CS*	*NE-CS*	*non-NE-CS*
Weight	5.4	1.58	1.83

Table 3: Weights assigned to features by the linear kernel SVM.

If this method of harnessing named entities and topic terms were indeed so effective, then we should have been able to detect every old story in every topic. However, analysis reveals that this approach makes an assumption about the way stories in a topic are related. Not all topics are *dense*, with both named entities and topic terms threading the stories together. Examples of such topics are natural disaster topics. While the first story might report on the actual calamity and the region it affected, successive stories might report on individual survivor tales. These stories might be connected to the original story of the topic by as tenuous a link as only the name of the calamity, or the place. Such topic structures are very common in newswire. Hence our approach will fail in such topics with loosely connected stories. Much more advanced processing of story content is required in such cases. Mistakes made by the named entity recognizer also impede performance.

Given that its impractical to expect manual transcriptions of all broadcast news, we tested our baseline and classifier systems on a version of TDT2 with newswire stories and ASR output of the broadcast news (*nwt*+*bn*-*asr*). TDT5 was left out as it doesn't have any broadcast news. As shown in Table 4, the baseline system performed significantly worse when manual transcription was replaced with ASR output. The classifier systems did even worse than the *nwt* +*bn*-*asr* baseline result. An analysis of the named entities extracted revealed that the accuracy was very poor - worse than extraction from *bn*-*man* documents. This was primarily because the version of IdentiFinder (IdentiFinder-*man*) we used was by default trained on *nwt*.

To alleviate this problem we re-trained IdentiFinder using a simulated ASR corpus with named entities identified correctly. Since the amount of training data required was huge, we obtained the training data from the *bn*-*man* version of TDT3. We ran IdentiFinder-*man* on the *bn*-*man* version of TDT3 and tagged the named entities. We then removed punctuation and converted all the text to uppercase to simulate ASR to a limited degree. We re-trained IdentiFinder on this simulated ASR corpus and used it to tag named entities in only the *bn*-*asr* stories in TDT2. We retained the use of IdentiFinder-*man* for the *nwt* stories. The same three features were then extracted and we re-ran the classifiers. The results are shown in Table 4 in the column titled IdentiFinder-*asr*.

Kernel Type	TDT2 (*nwt*+*bn*-*asr*)	
Baseline System	0.640	
	IdentiFinder-*man*	IdentiFinder-*asr*
Linear Kernel	0.653	**0.608**
Poly. of deg. 1	0.654	0.608
Poly. of deg. 2	0.658	0.619
Poly. of deg. 3	0.659	0.616
Poly. of deg. 4	0.671	0.632
Poly. of deg. 5	0.676	0.640
Poly. of deg. 6	0.682	0.652
RBF with γ = 1	0.649	0.636
RBF with γ = 5	0.668	0.679

Table 4: The baseline system was the same used for the *nwt*+*bn*-*man* collection. We find that using a linear kernel for the procedure using IdentiFinder-*asr* to tag named entities results in statistically significant improvement.

8 Conclusions and Future Work

We have shown the applicability of machine learning classification techniques to solve the NED problem. Significant improvements were made over the baseline systems on all the corpora tested on. The features we engineered made extensive use of named entities, and reinforced the importance and need to effectively harness their utility to solve problems in TDT. NED requires not only detection and reporting of new events, but also suppression of stories that report old events. From the study of the distributions of scores assigned to stories by the baseline and SVM model systems, we can see that we now do a better job of detecting old stories (reducing false alarms). Thus we believe that attacking the problem as "*old story detection*" might be a better and more fruitful approach. We have shown the effects of ASR output in the news stream, and demonstrated a procedure to alleviate the problem.

A classifier with RBF kernel with γ set to one exhibited the best performance. The reason for this superior performance over other kernels needs to be investigated. Engineering of better features is also a definite priority. In the future NED can also be extended to other interesting domains like scientific literature to detect the emerge of new topics and interests.

Acknowledgements
This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-9907018, and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

J. Allan, Hubert Jin, Martin Rajman, Charles Wayne, Daniel Gildea, Victor Lavrenko, Rose Hoberman, and David Caputo. 1999. Topic-based novelty detection. Technical report, Center for Language and Speech Processing, Johns Hopkins University. Summer Workshop Final Report.

J. Allan, Victor Lavrenko, and Hubert Jin. 2000. First story detection in tdt is hard. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 374–381. ACM Press.

J. Allan. 2002. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

Thorsten Brants, Francine Chen, and Ayman Farahat. 2003. A system for new event detection. In *Proceedings of the 26th Annual International ACM SIGIR Conference*, pages 330–337, New York, NY, USA. ACM Press.

Ronald K. Braun and Ryan Kaneshiro. 2003. Exploiting topic pragmatics for new event detection in tdt-2004. Technical report, National Institute of Standards and Technology. Topic Detection and Tracking Workshop.

Ronald K. Braun and Ryan Kaneshiro. 2004. Exploiting topic pragmatics for new event detection in tdt-2004. Technical report, National Institute of Standards and Technology. Topic Detection and Tracking Workshop.

Christopher J. C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

James P. Callan, W. Bruce Croft, and Stephen M. Harding. 1992. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83.

Ayman Farahat, Francine Chen, and Thorsten Brants. 2003. Optimizing story link detection is not equivalent to optimizing new event detection. In *ACL*, pages 232–239.

Thorsten Joachims. 1999. *Making large-scale support vector machine learning practical*. MIT Press, Cambridge, MA, USA.

Giridhar Kumaran and J. Allan. 2004. Text classifi cation and named entities for new event detection. In *Proceedings of the 27th Annual International ACM SIGIR Conference*, pages 297–304, New York, NY, USA. ACM Press.

Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. 2002. Applying semantic classes in event detection and tracking. In *Proceedings of International Conference on Natural Language Processing (ICON 2002)*, pages 175–183.

Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. 2004. Simple semantics in topic detection and tracking. *Information Retrieval*, 7(3–4):347–368.

Dunja Mladeni, Janez Brank, Marko Grobelnik, and Natasa Milic-Frayling. 2004. Feature selection using linear classifi er weights: interaction with classifi cation models. In *Proceedings of the 27th Annual International ACM SIGIR Conference*, pages 234–241, New York, NY, USA. ACM Press.

R. Papka and J. Allan. 1998. On-line new event detection using single pass clustering. Technical Report UM-CS-1998-021.

Nicola Stokes and Joe Carthy. 2001. Combining semantic and syntactic document classifi ers to improve fi rst story detection. In *Proceedings of the 24th Annual International ACM SIGIR Conference*, pages 424–425, New York, NY, USA. ACM Press.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference*, pages 28–36, New York, NY, USA. ACM Press.

Yiming Yang, Jian Zhang, Jaime Carbonell, and Chun Jin. 2002. Topic-conditioned novelty detection. In *Proceedings of the 8th ACM SIGKDD International Conference*, pages 688–693. ACM Press.

Jian Zhang, Zoubin Ghahramani, and Yiming Yang. 2005. A probabilistic model for online document clustering with application to novelty detection. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1617–1624. MIT Press, Cambridge, MA.

Investigating Unsupervised Learning for Text Categorization Bootstrapping

Alfio Gliozzo and **Carlo Strapparava**
ITC-irst
Istituto per la Ricerca Scientifica e Tecnologica
I-38050 Trento, Italy
{gliozzo,strappa}@itc.it

Ido Dagan
Computer Science Department
Bar Ilan University
Ramat Gan, Israel
dagan@cs.biu.ac.il

Abstract

We propose a generalized bootstrapping algorithm in which categories are described by relevant seed features. Our method introduces two unsupervised steps that improve the initial categorization step of the bootstrapping scheme: (i) using Latent Semantic space to obtain a generalized similarity measure between instances and features, and (ii) the Gaussian Mixture algorithm, to obtain uniform classification probabilities for unlabeled examples. The algorithm was evaluated on two Text Categorization tasks and obtained state-of-the-art performance using only the category names as initial seeds.

1 Introduction

Supervised classification is the task of assigning category labels, taken from a predefined set of categories (classes), to instances in a data set. Within the classical supervised learning paradigm, the task is approached by providing a learning algorithm with a training data set of manually labeled examples. In practice it is not always easy to apply this schema to NLP tasks. For example supervised systems for Text Categorization (TC) require a large amount of hand labeled texts, while in many applicative cases it is quite difficult to collect the required amounts of hand labeled data. Unlabeled text collections, on the other hand, are in general easily available.

An alternative approach is to provide the necessary supervision by means of sets of “seeds” of intuitively relevant features. Adopting terminology from computability theory, we refer to the standard example-based supervision mode as *Extensional Learning* (EL), as classes are being specified by means of examples of their elements (their *extension*). Feature-based supervision is referred to as *Intensional Learning* (IL), as features may often be perceived as describing the *intension* of a category, such as providing the name or prominent key terms for a category in text categorization.

The IL approach reflects on classical rule-based classification methods, where the user is expected to specify exact classification rules that operate in the feature space. Within the machine learning paradigm, IL has been incorporated as a technique for bootstrapping an extensional learning algorithm, as in (Yarowsky, 1995; Collins and Singer, 1999; Liu et al., 2004). This way the user does not need to specify exact classification rules (and feature weights), but rather perform a somewhat simpler task of specifying few typical seed features for the category. Given the list of seed features, the bootstrapping scheme consists of (i) preliminary unsupervised categorization of the unlabeled data set based on the seed features, and (ii) training an (extensional) supervised classifier using the automatic classification labels of step (i) as the training data (the second step is possibly reiterated, such as by an Expectation-Maximization schema). The core part of IL bootstrapping is step (i), i.e. the initial unsupervised classification of the unlabeled dataset. This step was often approached by relatively simple methods, which are doomed to obtain mediocre quality. Even so, it is hoped that the second step of supervised training would be robust enough to the noise in the initial training set.

The goal of this paper is to investigate additional principled unsupervised mechanisms within the initial classification step, applied to the text categorization. In particular, (a) utilizing a Latent Semantic Space to obtain better similarity assessments between seeds and examples, and (b) applying a Gaussian Mixture (GM) algorithm, which provides a principled unsupervised estimation of classification probability. As shown in our experiments, incorporating these steps consistently improved the accuracy of the initial categorization step, which in turn yielded a better final classifier thanks to the more accurate training set. Most importantly, we obtained comparable or better performance than previous IL methods using *only* the category names as the seed features; other IL methods required collecting a larger number of seed terms, which turns out to be a somewhat tricky task.

Interesting results were revealed when comparing our IL method to a state-of-the-art extensional classifier, trained on manually labeled documents. The EL classifier required 70 (Reuters dataset) or 160 (Newsgroup dataset) documents per category to achieve the same performance that IL obtained using only the category names. These results suggest that IL may provide an appealing cost-effective alternative when sub-optimal accuracy suffices, or when it is too costly or impractical to obtain sufficient labeled training. Optimal combination of extensional and intensional supervision is raised as a challenging topic for future research.

2 Bootstrapping for Text Categorization

The TC task is to assign category labels to documents. In the IL setting, a category C_i is described by providing a set of relevant features, termed an *intensional description* (ID), $id_{c_i} \subseteq V$, where V is the vocabulary. In addition a training corpus $T = \{t_1, t_2, \dots t_n\}$ of *unlabeled* texts is provided. Evaluation is performed on a separate test corpus of labeled documents, to which standard evaluation metrics can be applied.

The approach of categorizing texts based on lists of keywords has been attempted rather rarely in the literature (McCallum and Nigam, 1999; Ko and Seo, 2000; Liu et al., 2004; Ko and Seo, 2004). Several names have been proposed for it – such as *TC by bootstrapping with keywords*, *unsupervised TC*, *TC by labelling words* – where the proposed methods fall (mostly) within the IL settings described here[1].

It is possible to recognize a common structure of these works, based on a typical bootstrap schema (Yarowsky, 1995; Collins and Singer, 1999):

Step 1: *Initial unsupervised categorization.* This step was approached by applying some similarity criterion between the initial category seed and each unlabeled document. Similarity may be determined as a binary criterion, considering each seed keyword as a classification rule (McCallum and Nigam, 1999), or by applying an IR style vector similarity measure. The result of this step is an initial categorization of (a subset of) the unlabeled documents. In (Ko and Seo, 2004) term similarity techniques were exploited to expand the set of seed keywords, in order to improve the quality of the initial categorization.

Step 2: *Train a supervised classifier on the initially categorized set.* The output of Step 1 is exploited to train an (extensional) supervised classifier. Different learning algorithms have been tested, including SVM, Naive Bayes, Nearest Neighbors, and Rocchio. Some works (McCallum and Nigam, 1999; Liu et al., 2004) performed an additional Expectation Maximization algorithm over the training data, but reported rather small incremental improvements that do not seem to justify the additional effort.

(McCallum and Nigam, 1999) reported categorization results close to human agreement on the same task. (Liu et al., 2004) and (Ko and Seo, 2004) contrasted their word-based TC algorithm with the performance of an extensional supervised algorithm, achieving comparable results, while in general somewhat lower. It should be noted that it has been more difficult to define a common evaluation framework for comparing IL algorithms for TC, due to the subjective selection of seed IDs and to the lack of common IL test sets (see Section 4).

[1]The major exception is the work in (Ko and Seo, 2004), which largely follows the IL scheme but then makes use of labeled data to perform a chi-square based feature selection before starting the bootstrap process. This clearly falls outside the IL setting, making their results incomparable to other IL methods.

3 Incorporating Unsupervised Learning into Bootstrap Schema

In this section we show how the core Step 1 of the IL scheme – the initial categorization – can be boosted by two unsupervised techniques. These techniques fit the IL setting and address major constraints of it. The first is exploiting a generalized similarity metric between category seeds (IDs) and instances, which is defined in a Latent Semantic space. Applying such unsupervised similarity enables to enhance the amount of information that is exploited from each seed feature, aiming to reduce the number of needed seeds. The second technique applies the unsupervised Gaussian Mixture algorithm, which maps similarity scores to a principled classification probability value. This step enables to obtain a uniform scale of classification scores across all categories, which is typically obtained only through calibration over labeled examples in extensional learning.

3.1 Similarity in Latent Semantic Space

As explained above, Step 1 of the IL scheme assesses a degree of "match" between the seed terms and a classified document. It is possible first to follow the intuitively appealing and principled approach of (Liu et al., 2004), in which IDs (category seeds) and instances are represented by vectors in a usual IR-style Vector Space Model (VSM), and similarity is measured by the cosine function:

$$\mathbf{sim}_{vsm}(id_{c_i}, t_j) = \cos(\vec{id}_{c_i}, \vec{t_j}) \qquad (1)$$

where $\vec{id}_{c_i} \in \mathbf{R}^{|V|}$ and $\vec{t_j} \in \mathbf{R}^{|V|}$ are the vectorial representations in the space $\mathbf{R}^{|V|}$ respectively of the category ID id_{c_i} and the instance t_j, and V is the set of all the features (the vocabulary).

However, representing seeds and instances in a standard feature space is severely affected in the IL setting by feature sparseness. In general IDs are composed by short lists of features, possibly just a single feature. Due to data sparseness, most instances do not contain any feature in common with any category's ID, which makes the seeds irrelevant for most instances (documents in the text categorization case). Furthermore, applying direct matching only for a few seed terms is often too crude, as it ignores the identity of the other terms in the document.

The above problems may be reduced by considering some form of similarity in the feature space, as it enables to compare additional document terms with the original seeds. As mentioned in Section 2, (Ko and Seo, 2004) expanded explicitly the original category IDs with more terms, using a concrete query expansion scheme. We preferred using a generalized similarity measure based on representing features and instances a Latent Semantic (LSI) space (Deerwester et al., 1990). The dimensions of the Latent Semantic space are the most explicative principal components of the feature-by-instance matrix that describes the unlabeled data set. In LSI both coherent features (i.e. features that often co-occur in the same instances) and coherent instances (i.e. instances that share coherent features) are represented by similar vectors in the reduced dimensionality space. As a result, a document would be considered similar to a category ID if the seed terms and the document terms tend to co-occur overall in the given corpus.

The Latent Semantic Vectors for IDs and documents were calculated by an empirically effective variation (self-reference omitted for anonymity) of the *pseudo-document* methodology to fold-in documents, originally suggested in (Berry, 1992). The similarity function $\mathbf{sim}_{lsi}$ is computed by the cosine metric, following formula 1, where $\vec{id}_{c_i}$ and $\vec{t_j}$ are replaced by their Latent Semantic vectors. As will be shown in section 4.2, using such non sparse representation allows to drastically reduce the number of seeds while improving significantly the recall of the initial categorization step.

3.2 The Gaussian Mixture Algorithm and the initial classification step

Once having a similarity function between category IDs and instances, a simple strategy is to base the classification decision (of Step 1) directly on the obtained similarity values (as in (Liu et al., 2004), for example). Typically, IL works adopt in Step 1 a single-label classification approach, and classify each instance (document) to only one category. The chosen category is the one whose ID is most similar to the classified instance amongst all categories, which does not require any threshold tuning over labeled examples. The subsequent training in Step 2 yields a standard EL classifier, which can then be used to assign multiple categories to a document.

Using directly the output of the similarity function for classification is problematic, because the obtained scales of similarity values vary substantially across different categories. The variability in similarity value ranges is caused by variations in the number of seed terms per category and the levels of their generality and ambiguity. As a consequence, choosing the class with the highest absolute similarity value to the instance often leads to selecting a category whose similarity values tend to be generally higher, while another category could have been more similar to the classified instance if normalized similarity values were used.

As a solution we propose using an algorithm based on unsupervised estimation of Gaussian Mixtures (GM), which differentiates relevant and non-relevant category information using statistics from unlabeled instances. We recall that mixture models have been widely used in pattern recognition and statistics to approximate probability distributions. In particular, a well-known nonparametric method for density estimation is the so-called Kernel Method (Silverman, 1986), which approximates an unknow density with a mixture of kernel functions, such as gaussians functions. Under mild regularity conditions of the unknown density function, it can be shown that mixtures of gaussians converge, in a statistical sense, to *any* distribution.

More formally, let $t_i \in T$ be an instance described by a vector of features $\vec{t_i} \in \mathbf{R}^{|V|}$ and let $id_{c_i} \subset V$ be the ID of category C_i; let $\mathbf{sim}(id_{c_i}, t_j) \in \mathbf{R}$ be a similarity function among instances and IDs, with the only expectation that it monotonically increases according to the "closeness" of id_{c_i} and t_j (see Section 3.1).

For each category C_i, GM induces a mapping from the similarity scores between its ID and any instance t_j, $\mathbf{sim}(id_{c_i}, t_j)$, into the probability of C_i given the text t_j, $P(C_i|t_j)$. To achieve this goal GM performs the following operations: (i) it computes the set $\mathcal{S}_i = \{\mathbf{sim}(id_{c_i}, t_j)|t_j \in T\}$ of the similarity scores between the ID id_{c_i} of the category C_i and all the instances t_j in the unlabeled training set T; (ii) it induces from the empirical distribution of values in $\mathcal{S}_i$ a Gaussian Mixture distribution which is composed of two "hypothetic" distributions $\mathcal{C}_i$ and $\overline{\mathcal{C}_i}$, which are assumed to describe respectively the distributions of similarity scores for positive and negative examples; and (iii) it estimates the conditional probability $P(C_i|\mathbf{sim}(id_{c_i}, t_j))$ by applying the Bayes theorem on the distributions $\mathcal{C}_i$ and $\overline{\mathcal{C}}_i$. These steps are explained in more detail below.

The core idea of the algorithm is in step (ii). Since we do not have labeled training examples we can only obtain the set $\mathcal{S}_i$ which includes the similarity scores for all examples together, both positive and negative. We assume, however, that similarity scores that correspond to positive examples are drawn from one distribution, $P(\mathbf{sim}(id_{c_i}, t_j)|C_i)$, while the similarity scores that correspond to negative examples are drawn from another distribution, $P(\mathbf{sim}(id_{c_i}, t_j)|\overline{C_i})$. The observed distribution of similarity values in $\mathcal{S}_i$ is thus assumed to be a mixture of the above two distributions, which are recovered by the GM estimation.

Figure 1 illustrates the mapping induced by GM from the empirical mixture distribution: dotted lines describe the Probability Density Functions (PDFs) estimated by GM for $\mathcal{C}_i$, $\overline{\mathcal{C}_i}$, and their mixture from the empirical distribution ($\mathcal{S}_i$) (in step (ii)). The continuous line is the mapping induced in step (iii) of the algorithm from similarity scores between instances and IDs (x axis) to the probability of the instance to belong to the category (y axis).

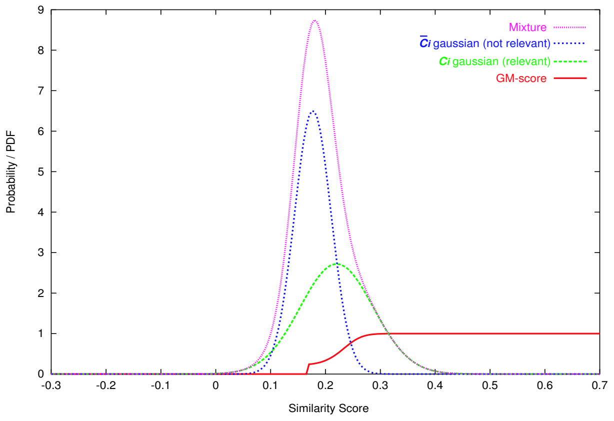


Figure 1: Mapping induced by GM for the category *rec.motorcycles* in the 20newsgroups data set.

The probabilistic mapping estimated in step (iii) for a category C_i given an instance t_j is computed by applying Bayes rule:

$$\begin{aligned} P(C_i|t_j) &= P(C_i|\mathrm{sim}(id_{c_i}, t_j)) = \\ &= \frac{P(\mathrm{sim}(id_{c_i}, t_j)|C_i)P(C_i)}{P(\mathrm{sim}(id_{c_i}, t_j)|C_i)P(C_i) + P(\mathrm{sim}(C_i, t_j)|\overline{C_i})P(\overline{C_i})} \end{aligned} \quad (2)$$

where $P(\mathbf{sim}(id_{c_i}, t_j)|C_i)$ is the value of the PDF of $\mathcal{C}_i$ at the point $\mathbf{sim}(id_{c_i}, t_j)$, $P(\mathbf{sim}(id_{c_i}, t_j)|\overline{C_i})$ is the value of the PDF of $\overline{\mathcal{C}_i}$ at the same point, $P(C_i)$ is the area of the distribution

$\mathcal{C}_i$ and $P(\overline{C_i})$ is the area of the distribution $\overline{\mathcal{C}_i}$. The mean and variance parameters of the two distributions $\mathcal{C}_i$ and $\overline{\mathcal{C}_i}$, used to evaluate equation 2, are estimated by the rather simple application of the Expectation Maximization (EM) algorithm for Gaussian Mixtures, as summarized in (Gliozzo et al., 2004).

Finally, following the single-labeled categorization setting of Step 1 in the IL scheme, the most likely category is assigned to each instance, that is, $argmax_{C_i} P(C_i|t_j)$.

3.3 Summary of the Bootstrapping Algorithm

step 1.a: Latent Semantic Space. Instances and Intensional Descriptions of categories (the seeds) are represented by vectors in Latent Semantic space. As an option, the algorithm can work with the classical Vector Space Model using the original feature space. Similarity scores between IDs and instances are computed by the Cosine measure.

step 1.b: GM. The mapping functions $P(C_i|t_j)$ for each category, conditioned on instances t_j, are induced by the GM algorithm. To that end, an Expectation Maximization algorithm estimates the parameters of the two component distributions of the observed mixture, which correspond to the distributions of similarity values for positive and negative examples. As an option, the GM mapping can be avoided.

step 1.c: Categorization. Each instance is classified to the most probable category - $argmax_{C_i} P(C_i|t_j)$.

step 2: Bootstrapping an extensional classifier. An EL classifier (SVM) is trained on the set of labeled instances resulting from step 1.c.

4 Evaluation

4.1 Intensional Text Categorization Datasets

Even though some typical data sets have been used in the TC literature (Sebastiani, 2002), the datasets used for IL learning were not standard. Often there is not sufficient clarity regarding details such as the exact version of the corpus used and the training/test splitting. Furthermore, the choice of categories was often not standard: (Ko and Seo, 2004) omitted 4 categories from the 20-Newsgroup dataset, while (Liu et al., 2004) evaluated their method on 4 separate subsets of the 20-Newsgroups, each containing only 4-5 categories. Such issues make it rather difficult to compare thoroughly different techniques, yet we have conducted several comparisons in Subsection 4.5 below. In the remainder of this Subsection we clearly state the corpora used in our experiments and the pre-processing steps performed on them.

20newsgroups. The 20 Newsgroups data set is a collection of newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. As suggested in the dataset Web site[2], we used the "bydate" version: the corpus (18941 documents) is sorted by date and divided in advance into a training (60%) set and a chronologically following test set (40%) (so there is no randomness in train/test set selection), it does not include cross-posts (duplicates), and (more importantly) does not include non-textual newsgroup-identifying headers which often help classification (Xref, Newsgroups, Path, Followup-To, Date).

We will first report results using *initial seeds* for the category ID's, which were selected using only the words in the category names, with some trivial transformations (i.e. `cryptography#n` for the category `sci.crypt`, `x-windows#n` for the category `comp.windows.x`). We also tried to avoid "overlapping" seeds, i.e. for the categories `rec.sport.baseball` and `rec.sport.hockey` the seeds are only {`baseball#n`} and {`hockey#n`} respectively and not {`sport#n, baseball#n`} and {`sport#n, hockey#n`}[3].

Reuters-10. We used the top 10 categories (Reuters-10) in the *Reuters-21578* collection Aptè split[4]. The complete Reuters collection includes 12,902 documents for 90 categories, with a fixed splitting between training and test data (70/30%). Both the Aptè and Aptè-10 splits are often used in TC tasks, as surveyed in (Sebastiani, 2002). To obtain the Reuters-10

[2]The collection is available at `www.ai.mit.edu/people/jrennie/20Newsgroups`.

[3]One could propose as a guideline for seed selection those seeds that maximize their distances in the LSI vector space model. On this perspective the LSI vectors built from {`sport#n, baseball#n`} and {`sport#n, hockey#n`} are closer than the vectors that represent {`baseball#n`} and {`hockey#n`}. It may be noticed that this is a reason for the slight initial performance decrease in the learning curve in Figure 2 below.

[4]available at `http://kdd.ics.uci.edu/databases/-reuters21578/reuters21578.html`).

Aptè split we selected the 10 most frequent categories: `Earn`, `Acquisition`, `Money-fx`, `Grain`, `Crude`, `Trade`, `Interest`, `Ship`, `Wheat` and `Corn`. The final data set includes 9296 documents. The initial seeds are only the words appearing in the category names.

Pre-processing. In both data sets we tagged the texts for part-of-speech and represented the documents by the frequency of each pos-tagged lemma, considering only nouns, verbs, adjectives, and adverbs. We induced the Latent Semantic Space from the training part[5] and consider the first 400 dimensions.

4.2 The impact of LSI similarity and GM on IL performance

In this section we evaluate the incremental impact of LSI similarity and the GM algorithm on IL performance. When avoiding both techniques the algorithm uses the simple cosine-based method over the original feature space, which can be considered as a baseline (similar to the method of (Liu et al., 2004)). We report first results using only the names of the categories as initial seeds.

Table 1 displays the F1 measure for the 20newsgroups and Reuters data sets, with and without LSI and with and without GM. The performance figures show the incremental benefit of both LSI and GM. In particular, when starting with just initial seeds and do not exploit the LSI similarity mechanism, then the performance is heavily penalized.

As mentioned above, the bootstrapping step of the algorithm (Step 2) exploits the initially classified instances to train a supervised text categorization classifier based on Support Vector Machines. It is worthwhile noting that the increment of performance after bootstrapping is generally higher when GM and LSI are incorporated, thanks to the higher quality of the initial categorization which was used for training.

4.3 Learning curves for the number of seeds

This experiment evaluates accuracy change as a function of the number of initial seeds. The experiment was performed for the 20 newsgroups corpus using both the LSI and the Classical vector space model. Additional seeds, beyond the category names, were identified by two lexicographers. For each category, the lexicographers were provided with a list of 100 seeds produced by the LSI similarity function applied to the category name (one list of 100 candidate terms for each category). From these lists the lexicographers selected the words that were judged as significantly related to the respective category, picking a mean of 40 seeds per category.

As seen in Figure 2, the learning curve using LSI vector space model dramatically outperforms the one using classical vector space. As can be expected, when using the original vector space (no generalization) the curve improves quickly with a few more terms. More surprisingly, with LSI similarity the best performance is obtained using the minimal initial seeds of the category names, while adding more seeds degrades performance. This might suggest that category names tend to be highly

[5] From a machine learning point of view, we could run the LSA on the full corpus (i.e. training and test), the LSA being a completely unsupervised technique (i.e. it does not take into account the data annotation). However, from an applicative point of view it is much more sensible to have the LSA built on the training part only. If we run the LSA on the full corpus, the performance figures increase in about 4 points.

			Reuters	*20 Newsgroups*
	LSI	*GM*	*F1*	*F1*
	no	no	0.38	0.25
+ *bootstrap*			0.42	0.28
	no	yes	0.41	0.30
+ *bootstrap*			0.46	0.34
	yes	no	0.46	0.50
+ *bootstrap*			**0.47**	**0.53**
	yes	yes	0.58	0.60
+ *bootstrap*			**0.74**	**0.65**

Table 1: Impact of LSI vector space and GM

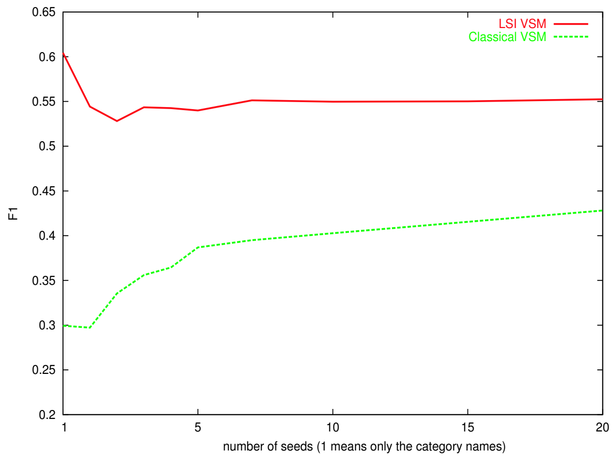


Figure 2: Learning curves on initial seeds for 20 newsgroups, LSI and Classical VSM (no LSI)

indicative for the intensional meaning of the category, and therefore adding more terms introduces additional noise. Further research is needed to find out whether other methods for selecting additional seed terms might yield incremental improvements. The current results, though, emphasize the benefit of utilizing LSI and GM. These techniques obtain state-of-the-art performance (see comparisons in Section 4.5) using only the category names as seeds, allowing us to skip the quite tricky phase of collecting manually a larger number of seeds.

4.4 Extensional vs. Intensional Learning

A major point of comparison between IL and EL is the amount of supervision effort required to obtain a certain level of performance. To this end we trained a supervised classifier based on Support Vector Machines, and draw its learning curves as a function of percentage of the training set size (Figure 3). In the case of 20newsgroups, to achieve the 65% F1 performance of IL the supervised settings requires about 3200 documents (about 160 texts per category), while our IL method requires only the category name. Reuters-10 is an easier corpus, therefore EL achieves rather rapidly a high performance. But even here using just the category name is equal on average to labeling 70 documents per-category (700 in total). These results suggest that IL may provide an appealing cost-effective alternative in practical settings when sub-optimal accuracy suffices, or when it is too costly or impractical to obtain sufficient amounts of labeled training sets.

It should also be stressed that when using the complete labeled training corpus state-of-the-art EL outperforms our best IL performance. This result deviates from the flavor of previous IL literature, which reported almost comparable performance relative to EL. As mentioned earlier, the method of (Ko and Seo, 2004) (as we understand it) utilizes labeled examples for feature selection, and therefore cannot be compared with our strict IL setting. As for the results in (Liu et al., 2004), we conjecture that their comparable performance for IL and EL may not be sufficiently general, for several reasons: the easier classification task (4 subsets of 20-Newsgroups of 4-5 categories each); the use of the usually weaker Naive-Bayes as the EL device; the use of clustering as an aid for selecting the seed terms from the 20-Newsgroup subsets, which might not scale up well when applied to a large number of categories of varying size.

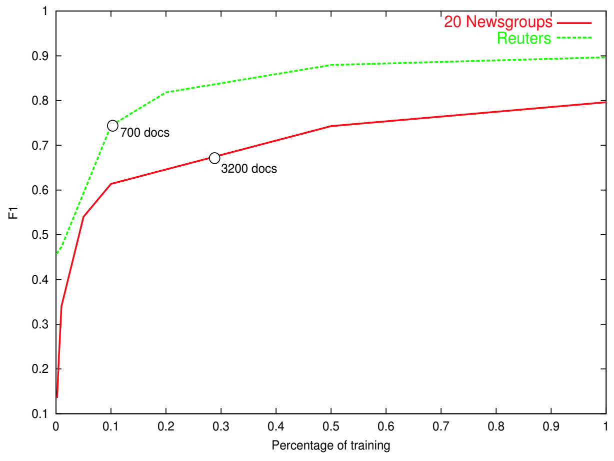


Figure 3: *Extensional* learning curves on as percentage of the training set.

4.5 Comparisons with other algorithms

As mentioned earlier it is not easy to conduct a thorough comparison with other algorithms in the literature. Most IL data sets used for training and evaluation are either not available (McCallum and Nigam, 1999) or are composed by somewhat arbitrary subsets of a standard data set. Another crucial aspect is the particular choice of the seed terms selected to compose an ID, which affects significantly the overall performance of the algorithm.

As a baseline system, we implemented a rule based approach in the spirit of (McCallum and Nigam, 1999). It is based on two steps. First, all the documents in the unlabeled training corpus containing at least one word in common with one and only one category ID are assigned to the respective class. Second, a supervised classifier based on SVM is trained on the labeled examples. Finally, the supervised classifier is used to perform the final categorization step on the test corpus. Table 2 reports the F1 measure of our replication of this method, using the category name as seed, which is substantially lower than the performance of the method we presented in this paper.

	Reuters	*20 Newsgroups*
	0.34	0.30
+ *bootstrap*	0.42	0.47

Table 2: Rule-based baseline performance

We also tried to replicate two of the non-standard data sets used in (Liu et al., 2004)[6]. Table 3 displays the performance of our approach in comparison to the results reported in (Liu et al., 2004). Following the evaluation metric adopted in that paper we report here accuracy instead of F1. For each data set (Liu et al., 2004) reported several results varying the number of seed words (from 5 to 30), as well as varying some heuristic thresholds, so in the table we report their best results. Notably, our method obtained comparable accuracy by using just the category name as ID for each class instead of multiple seed terms. This result suggests that our method enables to avoid the somewhat fuzzy process of collecting manually a substantial number of additional seed words.

	Our	*IDs per cat.*	*Liu et al.*	*IDs per cat.*
REC	0.94	1	0.95	5
TALK	0.80	1	0.80	20

Table 3: Accuracy on 4 “REC” and 4 “TALK” newsgroups categories

5 Conclusions

We presented a general bootstrapping algorithm for Intensional Learning. The algorithm can be applied to any categorization problem in which categories are described by initial sets of discriminative features and an unlabeled training data set is provided. Our algorithm utilizes a generalized similarity measure based on Latent Semantic Spaces and a Gaussian Mixture algorithm as a principled method to scale similarity scores into probabilities. Both techniques address inherent limitations of the IL setting, and leverage unsupervised information from an unlabeled corpus.

We applied and evaluated our algorithm on some text categorization tasks and showed the contribution of the two techniques. In particular, we obtain, for the first time, competitive performance using only the category names as initial seeds. This minimal information per category, when exploited by the IL algorithm, is shown to be equivalent to labeling about 70-160 training documents per-category for state of the art extensional learning. Future work is needed to investigate optimal procedures for collecting seed features and to find out whether additional seeds might still contribute to better performance. Furthermore, it may be very interesting to explore optimal combinations of intensional and extensional supervision, provided by the user in the forms of seed features *and* labeled examples.

[6]We used sequential splitting (70/30) rather than random splitting and did not apply any feature selection. This setting might be somewhat more difficult than the original one.

Acknowledgments

This work was developed under the collaboration ITC-irst/University of Haifa.

References

M. Berry. 1992. Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of EMNLP99*, College Park, MD, USA.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*.

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.

Y. Ko and J. Seo. 2000. Automatic text categorization by unsupervised learning. In *Proc. of COLING'2000*.

Y. Ko and J. Seo. 2004. Learning with unlabeled data for text categorization using bootstrapping abd feature projection techniques. In *Proc. of the ACL-04*, Barcelona, Spain, July.

B. Liu, X. Li, W. S. Lee, and P. S. Yu. 2004. Text classification by labeling words. In *Proc. of AAAI-04*, San Jose, July.

A. McCallum and K. Nigam. 1999. Text classification by bootstrapping with keywords, em and shrinkage. In *ACL99 - Workshop for Unsupervised Learning in Natural Language Processing*.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

B. W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL-95*, pages 189–196, Cambridge, MA.

Speeding up Training with Tree Kernels for Node Relation Labeling

Jun'ichi Kazama and **Kentaro Torisawa**
Japan Advanced Institute of Science and Technology (JAIST)
Asahidai 1-1, Nomi, Ishikawa, 923-1292 Japan
{kazama, torisawa}@jaist.ac.jp

Abstract

We present a method for speeding up the calculation of tree kernels during training. The calculation of tree kernels is still heavy even with efficient dynamic programming (DP) procedures. Our method maps trees into a small feature space where the inner product, which can be calculated much faster, yields the same value as the tree kernel for *most* tree pairs. The training is sped up by using the DP procedure only for the exceptional pairs. We describe an algorithm that detects such exceptional pairs and converts trees into vectors in a feature space. We propose tree kernels on *marked labeled ordered trees* and show that the training of SVMs for semantic role labeling using these kernels can be sped up by a factor of several tens.

1 Introduction

Many NLP tasks such as parse selection and tagging can be posed as the classification of labeled ordered trees. Several tree kernels have been proposed for building accurate kernel-based classifiers (Collins and Duffy, 2001; Kashima and Koyanagi, 2002). They have the following form in common.

$$K(T_1, T_2) = \sum_{S_i} W(S_i) \cdot \#_{S_i}(T_1) \cdot \#_{S_i}(T_2), \quad (1)$$

where S_i is a possible subtree, $\#_{S_i}(T_j)$ is the number of times S_i is *included* in T_j, and $W(S_i)$ is the weight of S_i. That is, tree kernels are inner products in a subtree feature space where a tree is mapped to vector $V(T_j) = \left(\sqrt{W(S_i)}\#_{S_i}(T_j)\right)_i$. With tree kernels we can take global structures into account, while alleviating overfitting with kernel-based learning methods such as support vector machines (SVMs) (Vapnik, 1995).

Previous studies (Collins and Duffy, 2001; Kashima and Koyanagi, 2002) showed that although it is difficult to explicitly calculate the inner product in Eq. (1) because we need to consider an exponential number of possible subtrees, the tree kernels can be computed in $O(|T_1||T_2|)$ time by using dynamic programming (DP) procedures. However, these DP procedures are time-consuming in practice.

In this paper, we present a method for speeding up the training with tree kernels. Our target application is *node relation labeling*, which includes NLP tasks such as semantic role labeling (SRL) (Gildea and Jurafsky, 2002; Moschitti, 2004; Hacioglu et al., 2004). For this purpose, we designed kernels on *marked labeled ordered trees* and derived $O(|T_1||T_2|)$ procedures. However, the lengthy training due to the cost of kernel calculation prevented us from assessing the performance of these kernels and motivated us to make the training practically fast.

Our speed-up method is based on the observation that very few pairs in the training set have a great many common subtrees (we call such pairs *malicious* pairs) and most pairs have a very small number of common subtrees. This leads to a drastic variance in kernel values, e.g., when $W(S_i) = 1$. We thus call this property of data *unbalanced similarity*. Fast calculation based on the inner product is possible for non-malicious pairs since we can convert the trees into vectors in a space of a small subset of all subtrees. We can speed up the training by using the DP procedure only for the rare malicious pairs.

We developed the FREQTM algorithm, a modification of the FREQT algorithm (Asai et al., 2002), to detect the malicious pairs and efficiently convert trees into vectors by enumerating only the subtrees actually needed (feature subtrees). The experiments demonstrated that our method makes the training of SVMs for the SRL task faster by a factor of several tens, and that it enables the performance of the kernels to be assessed in detail.

2 Kernels for Labeled Ordered Trees

The tree kernels proposed so far differ in how subtree inclusion is defined. For instance, Kashima and Koyanagi (2002) used the following definition.

DEFINITION 2.1 *S is included in T iff there exists a one-to-one function ψ from a node of S to a node of T such that (i) $pa(\psi(n_i)) = \psi(pa(n_i))$ ($pa(n_i)$ returns the parent of node n_i), (ii) $\psi(n_i) \succeq \psi(n_j)$ iff $n_i \succeq n_j$ ($n_i \succeq n_j$ means that n_i is an elder sibling of n_j), and (iii) $l(\psi(n_i)) = l(n_i)$ ($l(n_i)$ returns the label of n_i).*

We refer to the tree kernel based on this definition as K_{lo}. Collins and Duffy (2001) used a more restrictive definition where the preservation of CFG productions, i.e., $nc(\psi(n_i)) = nc(n_i)$ if $nc(n_i) > 0$ ($nc(n_i)$ is the number of children of n_i), is required in addition to the requirements in Definition 2.1. We refer to the tree kernel based on this definition as K_c.

It is pointed that extremely unbalanced kernel values cause overfitting. Therefore, Collins and Duffy (2001) used $W(S_i) = \lambda^{(\text{\# of productions in } S_i)}$, and Kashima and Koyanagi (2002) used $W(S_i) = \lambda^{|S_i|}$, where λ ($0 \leq \lambda \leq 1$) is a factor to alleviate the unbalance by penalizing large subtrees.

To calculate the tree kernels efficiently, Collins and Duffy (2001) presented an $O(|T_1||T_2|)$ DP procedure for K_c. Kashima and Koyanagi (2002) presented one for K_{lo}. The point of these procedures is that Eq. (1) can be transformed:

$$K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} C(n_1, n_2),$$
$$C(n_1, n_2) \equiv \textstyle\sum_{S_i} W(S_i) \cdot \#_{S_i}(T_1 \vartriangle n_1) \cdot \#_{S_i}(T_2 \vartriangle n_2),$$

where $\#_{S_i}(T_j \vartriangle n_k)$ is the number of times S_i is included in T_j with $\psi(root(S_i)) = n_k$. $C(n_1, n_2)$ can then be calculated recursively from those of the children of n_1 and n_2.

3 Kernels for Marked Labeled Ordered Trees for Node Relation Labeling

3.1 Node Relation Labeling

The node relation labeling finds relations among nodes in a tree. Figure 1 illustrates the concept of node relation labeling with the SRL task as an example. A0, A1, and AM-LOC are the semantic roles

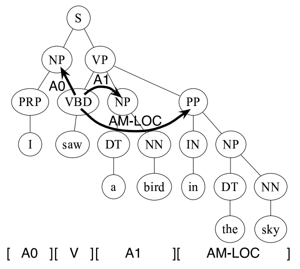


Figure 1: Node relation labeling.

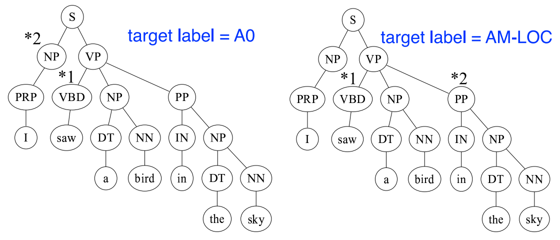


Figure 2: Semantic roles encoded by marked labeled ordered trees.

of the arguments of the verb "see (saw)". We represent an argument by the node that is the highest in the parse tree among the nodes that exactly cover the words in the argument. The node for the verb is determined similarly. For example, the node labeled "PP" represents the AM-LOC argument "in the sky", and the node labeled "VBD" represents the verb "see (saw)". We assume that there is a two-node relation labeled with the semantic role (represented by the arrow in the figure) between the verb node and the argument node.

3.2 Kernels on Marked Labeled Ordered Trees

We define a marked labeled ordered tree as a labeled ordered tree in which each node has a mark in addition to a label. We use $m(n_i)$ to denote the mark of node n_i. If n_i has no mark, $m(n_i)$ returns the special mark no-mark. We also use the function $marked(n_i)$, which returns $true$ iff $m(n_i)$ is not no-mark. We can encode a k-node relation by using k distinct marks. Figure 2 shows how the semantic roles illustrated in Figure 1 can be encoded using marked labeled ordered trees. We used the mark *1 to represent the verb node and *2 to represent the argument node.

The node relation labeling task can be posed as the classification of marked trees that returns $+1$ when the marks encode the correct relation and -1

Algorithm 3.1: KERNELLOMARK(T_1, T_2)

```
(nodes are ordered by the post-order traversal)
for $n_1 \leftarrow 1$ to $|T_1|$ do
  for $n_2 \leftarrow 1$ to $|T_2|$ do —————— (A)
    if $lm(n_1) \neq lm(n_2)$ then
      $C(n_1, n_2) \leftarrow 0$  $C^r(n_1, n_2) \leftarrow 0$
    else if $n_1$ and $n_2$ are leaf nodes then
      $C(n_1, n_2) \leftarrow \lambda$
      if $marked(n_1)$ and $marked(n_2)$ then
        $C^r(n_1, n_2) \leftarrow \lambda$  else $C^r(n_1, n_2) \leftarrow 0$
    else
      $S(0, j) \leftarrow 1$  $S(i, 0) \leftarrow 1$
      if $marked(n_1)$ and $marked(n_2)$ then
        $S^r(0, j) \leftarrow 1$  $S^r(i, 0) \leftarrow 1$
      else $S^r(0, j) \leftarrow 0$  $S^r(i, 0) \leftarrow 0$
      for $i \leftarrow 1$ to $nc(n_1)$ do
        for $j \leftarrow 1$ to $nc(n_2)$ do
          $S(i, j) \leftarrow$
            $S(i-1, j) + S(i, j-1) - S(i-1, j-1)$
            $+S(i-1, j-1) \cdot C(ch_i(n_1), ch_j(n_2))$
          $S^r(i, j) \leftarrow$ —————— (B)
            $S^r(i-1, j) + S^r(i, j-1) - S^r(i-1, j-1)$
            $+S^r(i-1, j-1) \cdot C(ch_i(n_1), ch_j(n_2))$
            $+S(i-1, j-1) \cdot C^r(ch_i(n_1), ch_j(n_2))$
            $-S^r(i-1, j-1) \cdot C^r(ch_i(n_1), ch_j(n_2))$
      $C(n_1, n_2) \leftarrow \lambda \cdot S(nc(n_1), nc(n_2))$
      $C^r(n_1, n_2) \leftarrow \lambda \cdot S^r(nc(n_1), nc(n_2))$
return $(\sum_{n_1=1}^{|T_1|} \sum_{n_2=1}^{|T_2|} C^r(n_1, n_2))$
```

otherwise. To enable such classification, we need tree kernels that take into account the node marks. We thus propose mark-aware tree kernels formulated as follows.

$$K(T_1, T_2) = \sum_{S_i : marked(S_i)} W(S_i) \cdot \#_{S_i}(T_1) \cdot \#_{S_i}(T_2),$$

where $marked(S_i)$ returns $true$ iff $marked(n_i) = true$ for at least one node in tree S_i. In these kernels, we require $m(\psi(n_i)) = m(n_i)$ in addition to $l(\psi(n_i)) = l(n_i)$ for subtree S_i to be regarded as included in tree T_j. In other words, these kernels treat $lm(n_i) \equiv (l(n_i), m(n_i))$ as the new label of node n_i and sum only over subtrees that have at least one marked node. We refer to the marked version of K_{lo} as K_{lo}^r and the marked version of K_c as K_c^r.

We can derive $O(|T_1||T_2|)$ DP procedures for the above kernels as well. Algorithm 3.1 shows the DP procedure for K_{lo}^r, which is derived by extending the DP procedure for K_{lo} (Kashima and Koyanagi, 2002). The key is the use of $C^r(n_1, n_2)$, which stores the sum over only marked subtrees, and its recursive calculation using $C(n_1, n_2)$ and $C^r(n_1, n_2)$ **(B)**. An $O(|T_1||T_2|)$ procedure for K_c^r can also be derived by extending (Collins and Duffy, 2001).

Table 1: Malicious and non-malicious pairs in the 1k data (3,136 trees) used in Sec. 5.2. We used $K(T_i, T_j) = 10^4$ with $\lambda = 1$ as the threshold for maliciousness. (A): pairs (i, i). (B): pairs from the same sentence except (i, i). (C): pairs from different sentences. Some malicious pairs are from different but similar sentences, which are difficult to detect.

		K_{lo}^r		K_c^r	
		# pairs	avg. $K(T_i, T_j)$	# of pairs	avg. $K(T_i, T_j)$
$\geq 10^4$	(A)	3,121	1.17×10^{52}	3,052	2.49×10^{32}
	(B)	7,548	7.24×10^{48}	876	1.26×10^{32}
	(C)	6,510	6.80×10^{9}	28	1.82×10^{4}
$< 10^4$	(A)	15	4.19×10^{3}	84	3.06×10^{3}
	(B)	4,864	2.90×10^{2}	11,536	1.27×10^{2}
	(C)	9,812,438	1.82×10^{1}	9,818,920	1.84×10^{-1}

4 Fast Training with Tree Kernels

4.1 Basic Idea

As mentioned, we define two types of tree pairs: malicious and non-malicious pairs. Table 1 shows how these two types of pairs are distributed in an actual training set. There is a clear distinction between malicious and non-malicious pairs, and we can exploit this property to speed up the training.

We define subset $\mathcal{F} = \{F_i\}$ (*feature subtrees*), which includes only the subtrees that appear as a common included subtree in the non-malicious pairs. We convert a tree to feature vector $V(T_j) = \left(\sqrt{W(F_i)}\#_{F_i}(T_j)\right)_i$ using only $\mathcal{F}$. Then we use a procedure that chooses the DP procedure or the inner product procedure depending on maliciousness:

$$K(T_i, T_j) = \begin{cases} K(T_i, T_j) \text{ (DP)} & \text{if } (i, j) \text{ is malicious.} \\ \langle V(T_i), V(T_j) \rangle & \text{otherwise} \end{cases}$$

This procedure returns the same value as the original calculation. Naively, if $|V(T_i)|$ (the number of feature subtrees such that $\#_{F_i}(T_i) \neq 0$) is small enough, we can expect a speed-up because the cost of calculating the inner product is $O(|V(T_i)| + |V(T_j)|)$. However, since $|V(T_i)|$ might increase as the training set becomes larger, we need a way to scale the speed-up to large data. In most kernel-based methods, such as SVMs, we actually need to calculate the kernel values with all the training examples for a given example T_i: $KS(T_i) = \{K(T_i, T_1), \ldots, K(T_i, T_L)\}$, where L is the number of training examples. Using *occurrence pattern* $OP(F_i) = \{(k, \#_{F_i}(T_k)) | \#_{F_i}(T_k) \neq 0\}$ pre-

```
Algorithm 4.1: CALCULATEKS($T_i$)

 for each $F$ such that $\#_F(T_i) \neq 0$ do
   for each $(j, \#_F(T_j)) \in OP(F)$ do
      $KS(j) \leftarrow KS(j) + W(F) \cdot \#_F(T_i) \cdot \#_F(T_j)$   (A)
 for $j = 1$ to $L$ do
     if $(i, j)$ is malicious then $KS(j) \leftarrow K(T_i, T_j)$  (DP)
```

```
Algorithm 4.2: FREQTM($D, R$)

 procedure GENERATECANDIDATE($F_i$)
  for each $(j, n) \in occ(F_i)$ do
    for each $(F_k, n_r) \in RME(F_i, T_j, n)$ do
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{F_k\}$;  $occ(F_k) \leftarrow occ(F_k) \cup (j, n_r)$
      if $|occ(F_k)|/|sup(F_i)| > R$ then
        return (($\phi$, false ))――――――――――――――(R)
  return (({$F_k | F_k \in \mathcal{S}, |sup(F_k)| \geq \underline{2}$}, true ))

 procedure SEARCH($F_i$, $\underline{precheck}$)
  if $|F_i| \geq D$ then REGISTERMAL($F_i$) return ( false )–(P)
  ($\mathcal{C}$, $\underline{suc}$) ← GENERATECANDIDATE($F_i$)
  if not $suc$ then REGISTERMAL($F_i$) return ( false )—(S)
  for each $F_k \in \mathcal{C}$ do
    if $malicious(F_k)$ then goto next $F_k$   ―――――-(P2)
    $suc$ ←SEARCH($F_k$, $precheck$)
    if not $suc$ and $|sup(F_i)| = |sup(F_k)|$ then
      return ( false )――――――――――――――――(P1)
  if not $precheck$ and $marked(F_i)$ then
    REGISTERSUBTREE($F_i$)――――――――――――――(F)
  return ( true )

 main
  $\mathcal{M} \leftarrow \phi$ (a set of malicious pairs)
  $\mathcal{F}^1 \leftarrow \{F_i \| F_i| = 1$ and $|sup(F_i)| \geq \underline{2}\}$
  for each $F_i \in \mathcal{F}^1$ do SEARCH($F_i$, true )――――――-(PC)
  for each $F_i \in \mathcal{F}^1$ do SEARCH($F_i$, false )
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,i) | 1 \leq i \leq l\}$
  return ($\underline{\mathcal{M}}$, $\{V(T_i)\}$, $\underline{\{W(f_i)\}}$)
```

Table 2: Functions in FREQTM.

- $occ(F_i)$ returns occurrence list of F_i whose element (j, n) indicates that F_i appears in T_j and that n (of T_j) is the node added to generated F_i in T_j by the RME (n works as the position of F_i in T_j).
- $sup(F_i)$ returns the IDs of distinct trees in $occ(F_i)$.
- $malicious(F_i)$ returns $true$ iff all pairs in $sup(F_i)$ are already registered in the set of malicious pairs, $\mathcal{M}$. (Currently, this returns $false$ if $|sup(F_i)| > M$ where M is the maximum support size of the malicious subtrees so far. We will remove this check since we found that it did not affect efficiency so much.)
- $RME(F_i, T_j, n)$ is a set of subtrees generated by RMEs of F_i in T_j (permitted when previously expanded node to generate F_i is n).

pared beforehand, we can calculate $KS(T_i)$ efficiently (Algorithm 4.1). A similar technique was used in (Kudo and Matsumoto, 2003a) to speed up the calculation of inner products.

We can show that the per-pair cost of Algorithm 4.1 is $O(c_1 Q + r_m c_2 |T_i||T_j|)$, where Q is the average number of common feature subtrees in a tree pair, r_m is the rate of malicious pairs, c_1 and c_2 are the constant factors for vector operations and DP operations. This cost is independent of the number of training examples. We expect from our observations that both Q and r_m are very small and that $c_1 \ll c_2$.

4.2 Feature Subtree Enumeration with Malicious Pair Detection

To detect malicious pairs and enumerate feature subtrees $\mathcal{F}$ (and to convert each tree to a feature vector), we developed an algorithm based on the FREQT algorithm (Asai et al., 2002). The FREQT algorithm can efficiently enumerate subtrees that are included (Definition 2.1) in more than a pre-specified number of trees in the training examples by generating candidate subtrees using *right most expansions* (*RMEs*). FREQT-based algorithms have recently been used in methods that treat subtrees as features (Kudo and Matsumoto, 2004; Kudo and Matsumoto, 2003b).

To develop the algorithm, we made the definition of maliciousness more search-oriented since it is costly to check for maliciousness based on the exact number of common subtrees or the kernel values (i.e., by using the DP procedure for all L^2 pairs). Whatever definition we use, the correctness is preserved as long as we do not fail to enumerate the subtrees that appear in the pairs we consider non-malicious. First, we consider pairs (i, i) to always be malicious. Then, we use a FREQT search that enumerates the subtrees that are included in at least two trees as a basis. Next, we modify FREQT so that it stops the search if candidate subtree F_i is too large (larger than size D, e.g., 20), and we regard the pairs of the trees where F_i appears as malicious because having a large subtree in common implies having a possibly exponential number of subtrees of that subtree in common. Although this test is heuristic and conservative in that it ignores the shape and marks of a tree, it works fine empirically.

Algorithm 4.2 is our algorithm, which we call FREQTM. The differences from FREQT are underlined. Table 2 summarizes the functions used. To make the search efficient, pruning is performed as follows (see also Figure 3). The basic idea behind is that if $malicious(F_i)$ is $true$ then $malicious(F_k)$ is also $true$ for F_k that is expanded from F_i by an

RME since $sup(F_k) \subseteq sup(F_i)$. This means we do not need to enumerate F_i nor any descendant of F_i.

- **(P)** Once $|F_i| \geq D$ and the malicious pairs are registered, we stop searching further.
- **(P1)** If the search from F_k (expanded from F_i) found a malicious subtree and if $|sup(F_i)| = |sup(F_k)|$, we stop the search from any other subtree F_m (expanded from F_i) since we can prove that $malicious(F_m) = true$ without actually testing it (proof omitted).
- **(P2)** If $malicious(F_k) = true$, we prune the search from F_k. To prune even when $malicious(F_k)$ becomes $true$ as a result of succeeding searches, we first run a search only for detecting malicious pairs (see **(PC)**).
- **(S)** We stop searching when the occurrence list becomes too long (larger than threshold R) since it causes a severe search slowdown.

Note that we use a depth-first version of FREQT as a basis to first find the largest subtrees and to detect malicious pairs at early points in the search. Enumeration of unnecessary subtrees is avoided because the registration of subtrees is performed at the post-order position **(F)**. The conversion to vectors is performed by assigning an ID to subtree F_i when registering it at **(F)** and distributing the ID to all the examples in $occ(F_i)$. Finally, D should be large enough to make r_m sufficiently small but should not be so large that too many feature subtrees are enumerated.

We expect that the cost of FREQTM is offset by the faster training, especially when training on the same data is repeatedly performed as in the tuning of hyperparameters.

For K_c^{τ}, we use a similar search procedure. However, the RME is modified so that all the children of a CFG production are expanded at once. Although the modification is not trivial, we omit the explanation due to space limitations.

4.3 Feature Compression

Additionally, we use a simple but effective feature compression technique to boost speed-up. The idea is simple: feature subtrees F_i and F_j can be treated as one feature f_k, with weight $W(f_k) = W(F_i) + W(F_j)$ if $OP(F_i) = OP(F_j)$. This drastically reduces the number of features. Although this is similar to finding closed and maximal subtrees (Chi et al., 2004), it is easy to implement since we need only the occurrence pattern, $OP(F_i)$, which is easily obtained from $occ(F_i)$ in the FREQTM search.

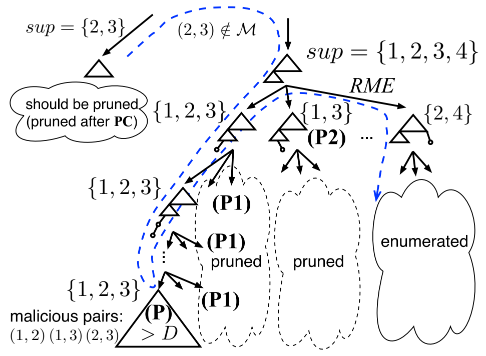


Figure 3: Pruning in FREQTM.

4.4 Alternative Methods

Vishwanathan and Smola (2004) presented the $O(|T_1| + |T_2|)$ procedure that exploits suffix trees to speed up the calculation of tree kernels. However, it can be applied to only a few types of subtrees that can be represented as a contiguous part in a string representation of a tree. Therefore, neither K_{lo}^{τ} nor K_c^{τ} can be sped up by using this procedure.

Another method is to change an inner loop, such as **(B)** in Algorithm 3.1, so that it iterates only over nodes in T_2 that have $l(n_1)$. We use this as the baseline for comparison, since we found that this is about two times faster than the standard implementation. [1]

4.5 Remaining Problem

Note that the method described here cannot speed up classification, since the converted vectors are valid only for calculating the kernels between trees in the training set. However, when we classify the same trees repeatedly, we can convert the trees in the training set and the classified trees at the same time and use the obtained vectors for classification.

5 Evaluation

We first evaluated the speed-up by our method for the semantic role labeling (SRL) task. We then demonstrated that the speed-up method enables a detailed comparison of K_{lo}^{τ} and K_c^{τ} for the SRL task.

[1]For K_c^{τ}, it might be possible to speed up comparisons in the algorithm by assigning IDs for CFG rules. We leave this for future work since it complicates implementation.

Table 3: Conversion statistics and speed-up for semantic role A2.

	K^r_{lo}					K^r_c				
size (# positive examples)	1,000	2,000	4,000	8,000	12,000	1,000	2,000	4,000	8,000	12,000
# examples	3,136	6,246	12,521	25,034	34,632	3,136	6,246	12,521	25,034	34,632
# feature subtrees ($\times 10^4$)	804.4	2,427.3	6,542.9	16,750.1	26,146. 5	3.473	9.009	21.867	52.179	78.440
# features (compressed) ($\times 10^4$)	20.7	67.3	207.2	585.9	977.0	0.580	1.437	3.426	8.128	12.001
avg. $\|V\|$ (compressed)	468.0	866.5	1,517.3	2,460.5	3,278.3	10.5	14.0	17.9	23.1	25.9
rate of malicious pairs r_m (%)	0.845	0.711	0.598	0.575	1.24	0.161	0.0891	0.0541	0.0370	0.0361
conversion time (sec.)	208.0	629.2	1,921.1	6,519.8	14,824.9	3.8	8.7	20.4	46.5	70.4
SVM time (DP+lookup) (sec.)	487.9	1,716.2	4,526.4	79,800.7	92,542.2	360.7	1,263.5	5,893.3	53,055.5	47,089.2
SVM time (proposed) (sec.)	17.5	68.6	186.4	1,721.7	2,531.8	4.9	25.7	119.5	982.8	699.1
speed-up factor	**27.8**	**25.0**	**24.3**	**46.4**	**36.6**	**73.3**	**49.1**	**49.3**	**53.98**	**67.35**

5.1 Setting

We used the data set provided for the CoNLL05 SRL shared task (Carreras and Màrquez, 2005). We used only the training part and divided it into our training, development, and testing sets (23,899, 7,966, and 7,967 sentences, respectively). As the tree structure, we used the output of Collins' parser (with WSJ-style non-terminals) provided with the data set. We also used POS tags by inserting the nodes labeled by POS tags above the word nodes. The average number of nodes in a tree was about 82. We ignored any arguments (and verbs) that did not match any node in the tree (the rate of such cases was about 3.5%). [2] The words were lowercased.

We used TinySVM[3] as the implementation of SVM and added our tree kernels, K^r_{lo} and K^r_c. We implemented FREQTM based on the implementation of FREQT by Kudo.[4] We normalized the kernel values: $K(T_i, T_j)/\sqrt{K(T_i, T_i) \times K(T_j, T_j)}$. Note that this normalization barely affected the training time since we can calculate $K(T_i, T_i)$ beforehand.

We assumed two-step labeling where we first find the argument node and then we determine the label by using a binary classifier for each semantic role. In this experiment, we focused on the performance for the classifiers in the latter step. We used the marked labeled ordered tree that encoded the target role as a positive example and the trees that encoded other roles of the verb in the same sentence as negative examples. We trained and evaluated the classifiers using the examples generated as above. [5]

[2]This was caused by parse errors, which can be solved by using more accurate parsers, and by bracketing inconsistencies between parser outputs and SRL annotations (e.g., phrasal verbs), many of which can be avoided by using heuristic transformers.

[3]http://chasen.org/˜taku/software/TinySVM

[4]http://chasen.org/˜taku/software/freqt

[5]The evaluation is slightly easier since the classifier for role X is evaluated only on the examples generated from the sentences that contain a verb that has X as a role.

5.2 Training Speed-up

We calculated the statistics for the conversion by FREQTM and measured the speed-up in SVM training for semantic role A2, for various numbers of training examples. For FREQTM, we used $D = 20$ and $R = 20$. For SVM training, we used convergence tolerance 0.001 (-e option in TinySVM), soft margin cost $C = 1.0 \times 10^3$ (-c), maximum number of iterations 10^5, kernel cache size 512 MB (-m), and decay factor $\lambda = 0.2$ for the weight of each subtree. We compared the time with our fast method (Algorithm 4.1) with that with the DP procedure with the node lookup described in Section 4.4. Note that these two methods yield almost identical SVM models (there are very slight differences due to the numerical computation). The time was measured using a computer with 2.4-GHz Opterons.

Table 3 shows the results for K^r_{lo} and K^r_c. The proposed method made the SVM training substantially faster for both K^r_{lo} and K^r_c. As we expected, the speed-up factor did not decrease even though $|V|$ increased as the amount of data increased. Note that FREQTM actually detected non-trivial malicious pairs such as those from very similar sentences in addition to trivial ones, e.g., (i, i). FREQTM conversion was much faster and the converted feature vectors were much shorter for K^r_c, presumably because K^r_c restricts the subtrees more.

The compression technique described in Section 4.3 greatly reduced the number of features. Without this compression, the storage requirement would be impractical. It also boosted the speed-up. For example, the training time with K^r_{lo} for the size 1,000 data in Table 3 was 86.32 seconds without compression. This means that the compression boosted the

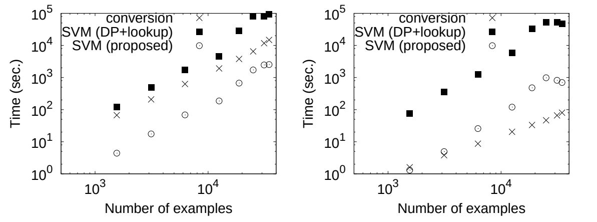


Figure 4: Scaling of conversion time and SVM training time. Left: K_{lo}^{r}. Right: K_{c}^{r}

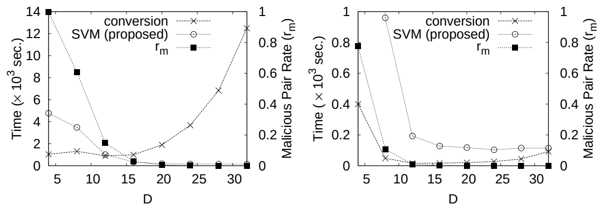


Figure 5: Relation between D and conversion time, SVM training time, and r_m. Left: K_{lo}^{r}. Right: K_{c}^{r}

speed-up by a factor of more than 5.

The cost of FREQTM is much smaller than that of SVM training with DP. Therefore, our method is beneficial even if we train the SVM only once.

To see how our method scales to large amounts of data, we plotted the time for the conversion and the SVM training w.r.t. data size on a log-log scale. As shown in Figure 4, the scaling factor was about 1.7 for the conversion time, 2.1 for SVM training with DP, and 2.0 for the proposed SVM training for K_{lo}^{r}. For K_{c}^{r}, the factors were about 1.3, 2.1, and 2.0, respectively. Regardless of the method, the cost of SVM training was about $O(L^2)$, as reported in the literature. Although FREQTM also has a super-linear cost, it is smaller than that of SVM training. Therefore, the cost of SVM training will become a problem before the cost of FREQTM does.

As we mentioned, the choice of D is a trade-off. Figure 5 shows the relationships between D and the time of conversion by FREQTM, the time of SVM training using the converted vectors, and the rate of malicious pairs, r_m. We can see that the choice of D is more important in the case of K_{lo} and that $D = 20$ used in our evaluation is not a bad choice.

5.3 Semantic Role Labeling

We assessed the performance of K_{lo}^{r} and K_{c}^{r} for semantic roles A1, A2, AM-ADV, and AM-LOC using our fast training method. We tuned soft margin cost C and λ by using the development set (we used the technique described in Section 4.5 to enable fast classification of the development set). We experimented with two training set sizes (4,000 and 8,000). For each λ (0.1, 0.15, 0.2, 0.25, and 0.30), we tested 40 different values of C ($C \in [2 \ldots 10^3]$ for size 4,000 and $C \in [0.5 \ldots 10^3]$ for size 8,000), and we evaluated the accuracy of the best setting for the test set.[6] Fast training is crucial since the performance differs substantially depending on the values of these hyperparameters. Table 4 shows the results. The accuracies are shown by F_1. We can see that K_{lo}^{r} outperformed K_{c}^{r} in all cases, presumably because K_{c}^{r} allows only too restrictive subtrees and therefore causes data sparseness. In addition, as one would expect, larger training sets are beneficial.

6 Discussion

The proposed speed-up method can also be applied to labeled ordered trees (e.g., for parse selection). However, the speed-up might be smaller since without node marks the number of subtrees increases while the DP procedure becomes simpler. On the other hand, the FREQTM conversion for marked labeled ordered trees might be made faster by exploiting the mark information for pruning. Although our method is not a complete solution in a classification setting, it might be in a clustering setting (in a sense it is training only). However, it is an open question whether unbalanced similarity, which is the key to our speed-up, is ubiquitous in NLP tasks and under what conditions our method scales better than the SVMs or other kernel-based methods.

Several studies claim that learning using tree kernels and other convolution kernels tends to overfit and propose selecting or restricting features (Cumby and Roth, 2003; Suzuki et al., 2004; Kudo and Matsumoto, 2004). Sometimes, the classification becomes faster as a result (Suzuki et al., 2004; Kudo and Matsumoto, 2004). We do not disagree with these studies. The fact that small λ values resulted in the highest accuracy in our experiment implies that too large subtrees are not so useful. However, since this tendency depends on the task, we need to assess the performance of full tree kernels for comparison. In this sense, our method is of great importance.

Node relation labeling is a generalization of node

[6]We used 10^6 as the maximum number of iterations.

Table 4: Comparison between K^r_{lo} and K^r_c.

		training set size = 4,000			training set size = 8,000		
		best setting	F_1 (dev)	F_1 (test)	best setting	F_1 (dev)	F_1 (test)
A1	K^r_{lo}	$\lambda = 0.2, C = 13.95$	**87.89**	**87.90**	$\lambda = 0.25, C = 8.647$	**89.80**	**89.81**
	K^r_c	$\lambda = 0.15, C = 3.947$	85.36	85.56	$\lambda = 0.2, C = 17.63$	87.93	87.96
A2	K^r_{lo}	$\lambda = 0.20, C = 13.95$	**85.65**	**84.70**	$\lambda = 0.20, C = 57.82$	**87.94**	**87.26**
	K^r_c	$\lambda = 0.10, C = 7.788$	84.79	83.51	$\lambda = 0.15, C = 1.0 \times 10^3$	87.37	86.23
AM-ADV	K^r_{lo}	$\lambda = 0.25, C = 8.647$	**86.20**	**86.64**	$\lambda = 0.15, C = 45.60$	**86.91**	**87.01**
	K^r_c	$\lambda = 0.20, C = 3.344$	83.58	83.72	$\lambda = 0.20, C = 2.371$	84.34	84.08
AM-LOC	K^r_{lo}	$\lambda = 0.15, C = 20.57$	**91.11**	**92.92**	N/A		
	K^r_c	$\lambda = 0.15, C = 13.95$	89.59	91.32	AM-LOC does not have more than 4,000 positive examples.		

marking where we determine the mark (tag) of a node. Kashima and Koyanagi (2002) dealt with this task by inserting the node representing the mark above the node to be tagged and classifying the transformed tree using SVMs with tree kernels such as K_{lo}. For the SRL task, Moschitti (2004) applied the tree kernel (K_c) to tree fragments that are heuristically extracted to reflect the role of interest. For relation extraction, Culotta and Sorensen (2004) proposed a tree kernel that operates on only the smallest tree fragment including two entities for which a relation is assigned. Our kernels on marked labeled ordered trees differ in what subtrees are permitted. Although comparisons are needed, we think our kernels are intuitive and general.

There are many possible structures for which tree kernels can be defined. Shen et al. (2003) proposed a tree kernel for LTAG derivation trees to focus only on linguistically meaningful structures. Culotta and Sorensen (2004) proposed a tree kernel for dependency trees. An important future task is to find suitable structures for each task (the SRL task in our case). Our speed-up method will be beneficial as long as there is unbalanced similarity.

7 Conclusion

We have presented a method for speeding up the training with tree kernels. Using the SRL task, we demonstrated that our speed-up method made the training substantially faster.

References

T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. 2002. Efficient substructure discovery from large semi-structured data. In *SIAM SDM'02*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *CoNLL 2005*.

Y. Chi, Y. Yang, Y. Xia, and R. R. Muntz. 2004. CMTreeMiner: Mining both closed and maximal frequent subtrees. In *PAKDD 2004*.

M. Collins and N. Duffy. 2001. Convolution kernels for natural language. In *NIPS 2001*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL 2004*.

C. Cumby and D. Roth. 2003. On kernel methods for relational learning. In *ICML 2003*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *CoNLL 2004*.

H. Kashima and T. Koyanagi. 2002. Kernels for semi-structured data. In *ICML 2002*, pages 291–298.

T. Kudo and Y. Matsumoto. 2003a. Fast methods for kernel-based text analysis. In *ACL 2003*.

T. Kudo and Y. Matsumoto. 2003b. Subtree-based Markov random fields and its application to natural language analysis (in Japanese). *IPSJ, NL-157*.

T. Kudo and Y. Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *EMNLP 2004*, pages 301–308.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *ACL 2004*.

L. Shen, A. Sarkar, and A. K. Joshi. 2003. Using LTAG based features in parse reranking. In *EMNLP 2003*.

J. Suzuki, H. Isozaki, and E. Maeda. 2004. Convolution kernels with feature selection for natural language processing tasks. In *ACL 2004*, pages 119–126.

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

S. V. N. Vishwanathan and A. J. Smola. 2004. Fast kernels for string and tree matching. *Kernels and Bioinformatics*.

Kernel-based Approach for Automatic Evaluation of Natural Language Generation Technologies: Application to Automatic Summarization

Tsutomu Hirao
NTT Communication Science Labs.
NTT Corp.
hirao@cslab.kecl.ntt.co.jp

Manabu Okumura
Precision and Intelligence Labs.
Tokyo Institute of Technology
oku@pi.titech.ac.jp

Hideki Isozaki
NTT Communication Science Labs.
NTT Corp.
isozaki@cslab.kecl.ntt.co.jp

Abstract

In order to promote the study of automatic summarization and translation, we need an accurate automatic evaluation method that is close to human evaluation. In this paper, we present an evaluation method that is based on convolution kernels that measure the similarities between texts considering their substructures. We conducted an experiment using automatic summarization evaluation data developed for Text Summarization Challenge 3 (TSC-3). A comparison with conventional techniques shows that our method correlates more closely with human evaluations and is more robust.

1 Introduction

Automatic summarization, machine translation, and paraphrasing have attracted much attention recently. These tasks include text-to-text language generation. Evaluation workshops are held in the U.S. and Japan, e.g., the Document Understanding Conference (DUC)[1], NIST Machine Translation Evaluation[2] as part of the TIDES project, the Text Summarization Challenge (TSC)[3] of the NTCIR project, and the International Workshop on Spoken Language Translation (IWSLT)[4].

These evaluation workshops employ human evaluations, which are essential in terms of achieving high quality evaluations results. However, human evaluations require a huge effort and the cost is considerable. Moreover, we cannot automatically evaluate a new system even if we use the corpora built for these workshops, and we cannot conduct re-evaluation experiments.

To cope with this situation, there is a particular need to establish a high quality automatic evaluation method. Once this is done, we can expect great progress to be made on natural language generation.

In this paper, we propose a novel automatic evaluation method for natural language generation technologies. Our method is based on the Extended String Subsequence Kernel (ESK) (Hirao et al., 2004b) which is a kind of convolution kernel (Collins and Duffy, 2001). ESK allows us to calculate the similarities between a pair of texts taking account of word sequences, their word sense sequences and their combinations.

We conducted an experimental evaluation using automatic summarization evaluation data developed for TSC-3 (Hirao et al., 2004a). The results of the comparison with ROUGE-N (Lin and Hovy, 2003; Lin, 2004a; Lin, 2004b), ROUGE-S(U) (Lin, 2004b; Lin and Och, 2004) and ROUGE-L (Lin, 2004a; Lin, 2004b) show that our method correlates more closely with human evaluations and is more robust.

2 Related Work

Automatic evaluation methods for automatic summarization and machine translation are grouped into two classes. One is the longest common subsequence (LCS) based approach (Hori et al., 2003; Lin, 2004a; Lin, 2004b; Lin and Och, 2004). The other is the N-gram based approach (Papineni et al.,

[1]http://duc.nist.gov
[2]http://www.nist.gov/speech/tests/mt/
[3]http://www.lr.titech.ac.jp/tsc
[4]http://www.slt.atr.co.jp/IWSLT2004

Table 1: Components of vectors corresponding to S1 and S2. Bold subsequences are common to S1 and S2.

d	subsequence	S1	S2	d	subsequence	S1	S2	d	subsequence	S1	S2
1	**Becoming**	1	1	2	**Becoming-is**	λ^2	λ^2	2	astronaut-DREAM	0	λ^2
	DREAM	1	1		**Becoming-my**	λ^3	λ^3		astronaut-ambition	0	λ^2
	SPACEMAN	1	1		***SPACEMAN-DREAM***	λ^3	λ^2		astronaut-is	0	1
	a	1	0		SPACEMAN-ambition	0	λ^2		astronaut-my	0	λ
	ambition	0	1		SPACEMAN-dream	λ^3	0		cosmonaut-DREAM	λ^3	0
	an	0	1		SPACEMAN-great	λ^2	0		cosmonaut-dream	λ^3	0
	astronaut	0	1		***SPACEMAN*-is**	1	1		cosmonaut-great	λ^2	0
	cosmonaut	1	0		***SPACEMAN*-my**	λ	λ		cosmonaut-is	1	0
	dream	1	0		a-DREAM	λ^4	0		cosmonaut-my	λ	0
	great	1	0		a-SPACEMAN	1	0		great-DREAM	1	0
	is	1	1		a-cosmonaut	1	0		great-dream	1	0
	my	1	1		a-dream	λ^4	0		**is-*DREAM***	λ^2	λ
2	**Becoming-*DREAM***	λ^5	λ^4		a-great	λ^3	0		is-ambition	0	λ
	Becoming-*SPACEMAN*	λ	λ		a-is	λ	0		is-dream	λ^2	0
	Becoming-a	1	0		a-my	λ^2	0		is-great	λ	0
	Becoming-ambition	0	λ^4		an-DREAM	0	λ^3		**is-my**	1	1
	Becoming-an	0	1		an-SPACEMAN	0	1		**my-*DREAM***	λ	1
	Becoming-astronaut	0	λ		an-ambition	0	λ^3		my-ambition	0	1
	Becoming-cosmonaut	λ	0		an-astronaut	0	1		my-dream	λ	0
	Becoming-dream	λ^5	0		an-is	0	λ		my-great	1	0
	Becoming-great	λ^4	0		an-my	0	λ^2				

2002; Lin and Hovy, 2003; Lin, 2004a; Lin, 2004b; Soricut and Brill, 2004).

Hori et. al (2003) proposed an automatic evaluation method for speech summarization based on word recognition accuracy. They reported that their method is superior to BLEU (Papineni et al., 2002) in terms of the correlation between human assessment and automatic evaluation. Lin (2004a; 2004b) and Lin and Och (2004) proposed an LCS-based automatic evaluation measure called ROUGE-L. They applied ROUGE-L to the evaluation of summarization and machine translation. The results showed that the LCS-based measure is comparable to N-gram-based automatic evaluation methods. However, these methods tend to be strongly influenced by word order.

Various N-gram-based methods have been proposed since BLEU, which is now widely used for the evaluation of machine translation. Lin et al. (2003) proposed a recall-oriented measure, ROUGE-N, whereas BLEU is precision-oriented. They reported that ROUGE-N performed well as regards automatic summarization. In particular, ROUGE-1, *i.e.*, unigram matching, provides the best correlation with human evaluation. Soricut et. al (2004) proposed a unified measure. They integrated a precision-oriented measure with a recall-oriented measure by using an extension of the harmonic mean formula. It performs well in evaluations of machine translation, automatic summarization, and question answering.

However, N-gram based methods have a critical problem; they cannot consider co-occurrences with gaps, although the LCS-based method can deal with them. Therefore, Lin and Och (2004) introduced skip-bigram statistics for the evaluation of machine translation. However, they did not consider longer skip-n-grams such as skip-trigrams. Moreover, their method does not distinguish between bigrams and skip-bigrams.

3 Kernel-based Automatic Evaluation

The above N-gram-based methods correlated closely with human evaluations. However, we think some skip-n-grams (n$\geq$3) are useful. In this paper, we employ the Extended String Subsequence Kernel (ESK), which considers both n-grams and skip-n-grams. In addition, the ESK allows us to add word senses to each word. The use of word senses enables flexible matching even when paraphrasing is used.

The ESK is a kind of convolution kernel (Collins and Duffy, 2001). Convolution kernels have recently attracted attention as a novel similarity measure in natural language processing.

3.1 ESK

The ESK is an extension of the String Subsequence Kernel (SSK) (Lodhi et al., 2002) and the Word Sequence Kernel (WSK) (Cancedda et al., 2003).

The ESK receives two node sequences as inputs and maps each of them into a high-dimensional vector space. The kernel's value is simply the inner product of the two vectors in the vector space. In order to discount long-skip-n-grams, the decay parameter λ is introduced.

We explain the computation of the ESK's value whose inputs are the sentences (S1 and S2) shown below. In the example, word senses are shown in braces.

S1 Becoming a cosmonaut:$\{$*SPACEMAN*$\}$ is my great dream:$\{$*DREAM*$\}$

S2 Becoming an astronaut:$\{$*SPACEMAN*$\}$ is my ambition:$\{$*DREAM*$\}$

In this case, "cosmonaut" and "astronaut" share the same sense $\{$*SPACEMAN*$\}$ and "ambition" and "dream" also share the same sense $\{$*DREAM*$\}$. We can use WordNet for English and *Goitaikei* (Ikehara et al., 1997) for Japanese.

Table 1 shows the subsequences derived from S1 and S2 and its weights. Note that the subsequence length is two or less. From the table, there are fifteen subsequences[5] that are common to S1 and S2. Therefore, $\mathrm{ESK}^{d=2}(\mathrm{S1},\mathrm{S2}) = 7 + \lambda + 2\lambda^2 + \lambda^3 + \lambda^4 + \lambda^5 + \lambda^6 + \lambda^9$. For reference, there are three unigrams, one bigram, zero trigrams and three skip-bigrams common to S1 and S2.

Formally, the ESK is defined as follows. T and U are node sequences.

$$\mathrm{ESK}^d(T,U) = \sum_{m=1}^{d} \sum_{t_i \in T} \sum_{u_j \in U} K_m(t_i,u_j) \tag{1}$$

$$K_m(t_i,u_j) = \begin{cases} val(t_i,u_j) & \text{if} \quad m=1 \\ K'_{m-1}(t_i,u_j)\cdot val(t_i,u_j) & \text{otherwise} \end{cases} \tag{2}$$

Here, d is the upper bound of the subsequence length and $K'_m(t_i,u_j)$ is defined as follows. t_i is the i-th node of T. u_j is the j-th node of U. The function $val(s,t)$ returns the number of attributes common to given nodes s and t.

$$K'_m(t_i,u_j) = \begin{cases} 0 \quad \text{if} \quad j=1 \\ \lambda K'_m(t_i,u_{j-1}) + K''_m(t_i,u_{j-1}) & \text{otherwise} \end{cases} \tag{3}$$

$K''_m(t_i,u_j)$ is defined as follows:

$$K''_m(t_i,u_j) = \begin{cases} 0 \quad \text{if} \quad i=1 \\ \lambda K''_m(t_{i-1},u_j) + K_m(t_{i-1},u_j). \end{cases} \tag{4}$$

[5]Bold subsequences in Table 1.

Finally, we define the similarity measure between T and U by normalizing ESK. This similarity can be regarded as an extension of the cosine measure.

$$\mathrm{Sim}^d_{\mathrm{esk}}(T,U) = \frac{\mathrm{ESK}^d(T,U)}{\sqrt{\mathrm{ESK}^d(T,T)\mathrm{ESK}^d(U,U)}}. \tag{5}$$

3.2 Automatic Evaluation based on ESK

Suppose, $\mathcal{C}$ is a system output, which consists of ℓ sentences, and $\mathcal{R}$ is a human written reference, which consists of m sentences. c_i is a sentence in $\mathcal{C}$, and r_j is a sentence in $\mathcal{R}$. We define two scoring functions for automatic evaluation. First, we define a precision-oriented measure as follows:

$$P^d_{\mathrm{esk}}(\mathcal{C},\mathcal{R}) = \frac{1}{\ell}\sum_{i=1}^{\ell} \max_{1\le j\le m} \mathrm{Sim}^d_{\mathrm{esk}}(c_i,r_j) \tag{6}$$

Symmetrically, we define a recall-oriented measure as follows:

$$R^d_{\mathrm{esk}}(\mathcal{C},\mathcal{R}) = \frac{1}{m}\sum_{j=1}^{m} \max_{1\le i\le \ell} \mathrm{Sim}^d_{\mathrm{esk}}(c_i,r_j) \tag{7}$$

Finally, we define a unified measure, *i.e.*, F-measure, as follows:

$$F^d_{\mathrm{esk}}(\mathcal{C},\mathcal{R}) = \frac{(1+\beta^2)\times R_{\mathrm{esk}}(\mathcal{C},\mathcal{R})\times P_{\mathrm{esk}}(\mathcal{C},\mathcal{R})}{R_{\mathrm{esk}}(\mathcal{C},\mathcal{R})+\beta^2\times P_{\mathrm{esk}}(\mathcal{C},\mathcal{R})} \tag{8}$$

β is a cost parameter for R_{esk} and P_{esk}. β's value is selected depending on the evaluation task. Since summary should not miss important information given in the human reference, recall is more important than precision. Therefore,a large β will yield good results.

3.3 Extension for Multiple References

When multiple human references (correct answers) are available, we define a simple function for multiple references as follows:

$$F^{\mathrm{mean}}_{\mathrm{esk}}(\mathcal{C},R) = \frac{1}{n}\sum_{i=1}^{n} F_{\mathrm{esk}}(\mathcal{C},\mathcal{R}_i), \tag{9}$$

Here, equation (9) gives the average score. R indicates a set of references; $R = \{\mathcal{R}_1,\cdots,\mathcal{R}_n\}$.

4 Experimental Evaluation

To confirm and discuss the effectiveness of our method, we conducted an experimental evaluation using TSC-3 multiple document summarization evaluation data and our additional data.

4.1 Task and Evaluation Metrics in TSC-3

The task of TSC-3 is multiple document summarization. Participants were given a set of documents about a certain event and required to generate two different length summaries for the entire document set. The lengths were about 5% and 10% of the total number of characters in the document set, respectively. Thirty document sets were provided for the official run evaluation. There were ten participant systems; one provided by the TSC organizers as a baseline system.

The evaluation metric follows DUC's SEE evaluation scheme (Harman and Over, 2004). For each document set, one human subject makes a reference summary and uses it as a basis for evaluating ten system outputs. This human evaluation procedure consists of the following steps:

Step 1 For each reference sentence $r_j(\in \mathcal{R})$, repeat Steps 2 and 3.

Step 2 For r_j, the human assessor finds the most relevant sentence set S from the system output.

Step 3 The assessor assigns a score, $e(r_j, S)$, $0, 0.1, \cdots, 1.0$. 1.0 means perfect. in terms of how much of the content of r_j can be reproduced by using only sentences in S.

Step 4 Finally, the evaluation score of output $\mathcal{C}$ for reference $\mathcal{R}$ is defined $H(\mathcal{R}, \mathcal{C}) = \sum_j e(r_j, S)/|\mathcal{R}|$.

The final score of a system is calculated by applying the above procedure and normalized by the number of topics, *i.e.*, $\sum_{t=1}^{30} H(\mathcal{R}_t, \mathcal{C}_t)/30$. When multiple references $R(=\{\mathcal{R}1, \cdots, \mathcal{R}_n\})$ are available, the scores are given as follows: $H^{\text{mean}}(R, \mathcal{C}) = \sum_k H(\mathcal{R}_k, \mathcal{C})/|R|$.

4.2 Variation of Human Assessors

In TSC-3's official run evaluation, system outputs were compared with one human written reference summary for each topic. There were five topic sets and five human assessors (A-E in Table 2) for each topic set.

Before we use the one human written reference summary as the gold-standard-reference, to examine variations among human assessors, we prepared two additional human summaries for each topic sets.

Table 2: The relationship between topics and reference summary creators, *i.e.*, human assessors. $S(A)$ indicates a subject A's evaluation score for all systems for corresponding topics.

topic-ID	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{avg}$
1 - 6	S(A)	S(E)	S(C)	mean(S(A),S(E),S(C))
7 - 12	S(B)	S(A)	S(D)	mean(S(B),S(A),S(D))
13 - 18	S(C)	S(B)	S(E)	mean(S(C),S(B),S(E))
19 - 24	S(D)	S(C)	S(A)	mean(S(D),S(C),S(A))
25 - 30	S(E)	S(D)	S(B)	mean(S(E),S(D),S(B))

Table 3: Correlations between human judgments.

	correlation coefficient (r)				rank correlation coefficient (ρ)			
short								
	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{\text{avg}}$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{\text{avg}}$
$\mathcal{D}_1$	1.00	.968	.902	.988	1.00	.976	.697	.988
$\mathcal{D}_2$	–	1.00	.910	.996	–	1.00	.733	.988
$\mathcal{D}_3$	–	–	1.00	.914	–	–	1.00	.758
$\mathcal{D}_{\text{avg}}$	–	–	–	1.00	–	–	–	1.00
long								
	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{\text{avg}}$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{\text{avg}}$
$\mathcal{D}_1$	1.00	.908	.822	.964	1.00	.964	.939	.964
$\mathcal{D}_2$	–	1.00	.963	.987	–	1.00	.952	1.00
$\mathcal{D}_3$	–	–	1.00	.931	–	–	1.00	.932
$\mathcal{D}_{\text{avg}}$	–	–	–	1.00	–	–	–	1.00

Therefore, we obtained three reference summaries and evaluation results for each topic sets (Table 2).

Moreover, we prepared unified evaluation results of three human judgment as $\mathcal{D}_{\text{avg}}$, which is calculated as the average of three human scores.

The relationship between topics and human assessors is shown in Table 2. For example, subject B generates summaries and evaluates all systems for topics 7-12, 13-18 and 25-30 on $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ respectively. Note that each human subject, A to E, was a retired professional journalist; that is, they shared a common background.

Table 3 shows the Pearson's correlation coefficient (r) and Spearman's rank correlation coefficient ρ for the human subjects. The results show that every pair has a high correlation. Therefore, changing the human subject has little influence as regards creating references and evaluating system summaries. The evaluation by human subjects is stable. This result agrees with DUC's additional evaluation results (Harman and Over, 2004). However, the behavior of the correlations between humans with different backgrounds is uncertain. The correlation might be fragile if we introduce a human subject whose background is different from the others.

4.3 Compared Automatic Evaluation Methods

We compared our method with ROUGE-N and ROUGE-L described below. We used only content words to calculate the ROUGE scores because the correlation coefficient decreased if we did not remove functional words.

WSK-based method

We use WSK instead of ESK in equation (6)-(8).

ROUGE-N

ROUGE-N is an N-gram-based evaluation measure defined as follows (Lin, 2004b):

$$\text{ROUGE-N}(\mathcal{C},\mathcal{R})=\frac{\sum_{\mathcal{S}\in\mathcal{R}}\sum_{\text{gram}_\text{N}\in\mathcal{S}}\text{Count}_\text{match}(\text{gram}_\text{N})}{\sum_{\mathcal{S}\in\mathcal{R}}\sum_{\text{gram}_\text{N}\in\mathcal{S}}\text{Count}(\text{gram}_\text{N})} \quad (10)$$

Here, $\text{Count}(\text{gram}_\text{N})$ is the number of an N-gram and $\text{Count}_\text{match}(\text{gram}_\text{N})$ denotes the number of n-gram co-occurrences in a system output and the reference.

ROUGE-S

ROUGE-S is an extension of ROUGE-2 defined as follows (Lin, 2004b):

$$\text{ROUGE-S}(\mathcal{C},\mathcal{R})=\frac{(1+\beta^2)\times R_\text{skip2}(\mathcal{C},\mathcal{R})\times P_\text{skip2}(\mathcal{C},\mathcal{R})}{R_\text{skip2}(\mathcal{C},\mathcal{R})+\beta^2 P_\text{skip2}(\mathcal{C},\mathcal{R})} \quad (11)$$

Where R_skip2 and P_skip2 are defined as follows:

$$\text{R}_\text{skip2}(\mathcal{C},\mathcal{R})=\frac{\text{Skip2}(\mathcal{C},\mathcal{R})}{\text{\# of skip bigram} \in \mathcal{R}} \quad (12)$$

$$\text{P}_\text{skip2}(\mathcal{C},\mathcal{R})=\frac{\text{Skip2}(\mathcal{C},\mathcal{R})}{\text{\# of skip bigram} \in \mathcal{C}} \quad (13)$$

Here, function Skip2 returns the number of skip-bi-grams that are common to $\mathcal{R}$ and $\mathcal{C}$.

ROUGE-SU

ROUGE-SU is an extension of ROUGE-S, which includes unigrams as a feature defined as follows (Lin, 2004b):

$$\text{ROUGE-SU}(\mathcal{C},\mathcal{R})=\frac{(1+\beta^2)\times R_\text{su}(\mathcal{C},\mathcal{R})\times P_\text{su}(\mathcal{C},\mathcal{R})}{R_\text{su}(\mathcal{C},\mathcal{R})+\beta^2 P_\text{su}(\mathcal{C},\mathcal{R})} \quad (14)$$

Where R_su and P_su are defined as follows:

$$\text{R}_\text{su}(\mathcal{C},\mathcal{R})=\frac{\text{SU}(\mathcal{C},\mathcal{R})}{\text{(\# of skip bigrams + \# of unigrams)} \in \mathcal{R}} \quad (15)$$

$$\text{P}_\text{su}(\mathcal{C},\mathcal{R})=\frac{\text{SU}(\mathcal{C},\mathcal{R})}{\text{(\# of skip bigrams + \# of unigrams)} \in \mathcal{C}} \quad (16)$$

Here, function SU returns the number of skip-bi-grams and unigrams that are common to $\mathcal{R}$ and $\mathcal{C}$.

ROUGE-L

ROUGE-L is an LCS-based evaluation measure defined as follows (Lin, 2004b):

$$\text{ROUGE-L}(\mathcal{C},\mathcal{R})=\frac{(1+\beta^2)\times R_\text{lcs}(\mathcal{C},\mathcal{R})\times P_\text{lcs}(\mathcal{C},\mathcal{R})}{R_\text{lcs}(\mathcal{C},\mathcal{R})+\beta^2 P_\text{lcs}(\mathcal{C},\mathcal{R})} \quad (17)$$

where R_lcs and P_lcs are defined as follows:

$$R_\text{lcs}(\mathcal{C},\mathcal{R})=\frac{1}{u}\sum_{r_i\in\mathcal{R}}\text{LCS}_\cup(r_i,\mathcal{C}) \quad (18)$$

$$P_\text{lcs}(\mathcal{C},\mathcal{R})=\frac{1}{v}\sum_{r_i\in\mathcal{R}}\text{LCS}_\cup(r_i,\mathcal{C}) \quad (19)$$

Here, $\text{LCS}_\cup(r_i,\mathcal{C})$ is the LCS score of the union longest common subsequence between reference sentences r_i and $\mathcal{C}$. u and v are the number of words contained in $\mathcal{R}$, and $\mathcal{C}$, respectively.

The multiple reference version of ROUGE-N S, SU or L, RN^mean, RS^mean, RSU^mean, RL^mean can be defined in accordance with equation (9).

4.4 Evaluation Measures

We evaluate automatic evaluation methods by using Pearson's correlation coefficient (r) and Spearman's rank correlation coefficient (ρ). Since we have ten systems, we make a vector $\mathbf{x}=(x_1,x_2,\cdots,x_i,\cdots,x_{10})$ from the results of an automatic evaluation. Here, $x_i=1/30\sum_{t=1}^{30} f(\mathcal{R}_t,\mathcal{C}_{i,t})$. $\mathcal{R}_t$ indicates a reference for the t-th topic. f indicates an automatic evaluation function such as F_esk, F_wsk, ROUGE-N, ROUGE-S, ROUGE-SU and ROUGE-L. Next, we make another vector $\mathbf{y}=(y_1,y_2,\cdots,y_i,\cdots,y_{10})$ from the human evaluation results. Here, $y_i=1/30\sum_{t=1}^{30} H(\mathcal{R}_t,\mathcal{C}_{i,t})$. Finally, we compute r and ρ between $\mathbf{x}$ and $\mathbf{y}$[6].

4.5 Evaluation Results and Discussions

Table 4 shows the evaluation results obtained by using Pearson's correlation coefficient r. Table 5 shows the evaluation results obtained with Spearman's rank correlation coefficient ρ. The ta-

[6]When using multiple references, functions f and H for making vectors $\mathbf{x}$ and $\mathbf{y}$ are substituted for f^mean and H^mean, respectively.

Table 4: Results obtained with Pearson's correlation coefficient."stop" indicates with stop word exclusion, "case" indicates w/o stop word exclusion.

	short								long							
	$\mathcal{D}_1$		$\mathcal{D}_2$		$\mathcal{D}_3$		$\mathcal{D}_{\text{avg}}$		$\mathcal{D}_1$		$\mathcal{D}_2$		$\mathcal{D}_3$		$\mathcal{D}_{\text{avg}}$	
	stop	case	stop	case	stop	case	stop	case	stop	case	stop	case	stop	case	stop	case
ROUGE-1	**.965**	.884	.931	.888	.937	.879	.956	.906	.906	.876	.919	.916	.897	.891	.918	.948
ROUGE-2	.943	.960	.836	.880	.861	.906	.904	.937	.886	.930	.788	.941	.834	.616	.856	.929
ROUGE-3	.906	.936	.759	.814	.786	.846	.862	.900	.873	.909	.717	.849	.826	.431	.844	.885
ROUGE-4	.878	.914	.725	.752	.729	.794	.837	.871	.850	.890	.651	.787	.836	.292	.836	.865
ROUGE-L	.919	.777	.789	.683	.875	.867	.898	.852	.917	.840	.861	.812	.847	.829	.910	.848
ROUGE-S(∞)	.934	.914	.805	.888	.872	**.938**	.867	.917	.812	.863	.744	.954	.709	.547	.757	.900
ROUGE-S(9)	.929	.935	.783	.899	.808	.917	.856	.939	.840	.903	.735	.951	.730	.617	.787	.927
ROUGE-S(4)	.936	.943	.802	.891	.839	.917	.877	.940	.876	.920	.778	.945	.814	.663	.840	.932
ROUGE-SU(∞)	.934	.914	.805	.887	.872	.937	.867	.917	.811	.864	.743	.954	.707	.547	.756	.900
ROUGE-SU(9)	.926	.938	.765	.890	.789	.906	.845	.936	.829	.904	.705	.948	.701	.586	.766	.925
ROUGE-SU(4)	.930	.945	.772	.865	.810	.889	.861	.927	.868	.921	.730	.928	.785	.620	.818	.925
$F_{\text{esk}}^{d=2}(\beta{=}2)$	.942		.927		.921		.957		**.941**		.957		**.967**		**.969**	
$F_{\text{esk}}^{d=2}(\beta{=}3)$	.929		**.943**		.928		**.965**		.939		**.962**		.959		.967	
$F_{\text{esk}}^{d=3}(\beta{=}2)$	.939		.923		.919		.962		.926		.954		.953		.966	
$F_{\text{esk}}^{d=3}(\beta{=}3)$	.927		.933		.920		.964		.920		.947		.904		.949	
$F_{\text{esk}}^{d=4}(\beta{=}2)$	.921		.900		.897		.955		.900		.932		.890		.946	
$F_{\text{esk}}^{d=4}(\beta{=}3)$	.909		.900		.888		.950		.892		.921		.819		.922	
$F_{\text{wsk}}^{d=2}(\beta{=}2)$	.939		.900		.897		.942		.931		.923		.936		.939	
$F_{\text{wsk}}^{d=2}(\beta{=}3)$	.928		.921		.909		.958		.932		.939		.950		.950	
$F_{\text{wsk}}^{d=3}(\beta{=}2)$	.938		.902		.886		.947		.924		.921		.934		.944	
$F_{\text{wsk}}^{d=3}(\beta{=}3)$	.928		.922		.895		.960		.920		.929		.919		.942	
$F_{\text{wsk}}^{d=4}(\beta{=}2)$	.929		.896		.873		.947		.910		.913		.908		.938	
$F_{\text{wsk}}^{d=4}(\beta{=}3)$	.918		.915		.879		.956		.903		.913		.865		.925	

bles show results obtained with and without stop word exclusion for the entire ROUGE family. For ROUGE-S and ROUGE-SU, we use three variations following (Lin, 2004b): the maximum skip distances are 4, 9 and infinity [7]. In addition, we examine $\beta = 2 \text{and} 3$ for the ESK-based and WSK-based methods. The decay parameter λ for F_{esk} and F_{wsk} is set at 0.5. We will discuss these parameter values in Section 4.6.

From the tables, ROUGE-N's r and ρ decrease monotonically with N when we exclude stop words. In most cases, the performance is improved by including stop words for N ($\geq$2). There is a large difference between ROUGE-1 and ROUGE-4. The ROUGE-S family is comparable to the ROUGE-SU family and their performance is close to ROUGE-1 without stop words and ROUGE-2 with stop words. ROUGE-L is better than both ROUGE-3 and ROUGE-4 but worse than ROUGE-1 or ROUGE-2.

On the other hand, F_{esk}'s correlation coefficients (r) do not change very much with respect to d. Even if d is set at 4, we can obtain good correlations. The behavior of rank correlation coefficients (ρ) is similar to the above. The difference between the ROUGE family and our method is particularly large for long summaries. By setting $d{=}2$, our method gives the good results. The optimal β is varied in the data sets. However, the difference between $\beta = 2$ and $\beta{=}3$ is small.

For ρ, our method outperforms the ROUGE family except for $\mathcal{D}_1$. By contrast, we can see $d{=}3$ or $d{=}4$ provided the best results. The differences between our method and the ROUGE family are larger than for r.

For both r and ρ, when multiple references are available, our method outperforms the ROUGE family.

Although ROUGE-1 sometimes provides better results than our method for short summaries, it has a critical problem; ROUGE-1 disregards word sequences making it easy to cheat. For instance, we can easily obtain a high ROUGE-1 score by using a sequence of high Inverse Document Frequency (IDF) words. Such a summary is incomprehensible and meaningless but we obtain a good ROUGE-1 score comparable to those of the top TSC-3 systems. By contrast, it is difficult to cheat other members of the ROUGE family or our method.

Our evaluation results imply that F_{esk} is robust

[7]We use β=1,2, and 3. However there are little difference among correlation coefficient regardless of β because the number of the words in reference and the number of the words in system output are almost the same.

Table 5: Results obtained with Spearman's correlation coefficient. "stop" indicates with stop word exclusion, "case" indicates w/o stop word exclusion.

	short								long							
	$\mathcal{D}_1$		$\mathcal{D}_2$		$\mathcal{D}_3$		$\mathcal{D}_{\rm avg}$		$\mathcal{D}_1$		$\mathcal{D}_2$		$\mathcal{D}_3$		$\mathcal{D}_{\rm avg}$	
	stop	case	stop	case	stop	case	stop	case	stop	case	stop	case	stop	case	stop	case
ROUGE-1	**.988**	.964	.842	.891	.842	.855	.927	.903	.818	.830	.903	.806	.867	.855	.842	**.915**
ROUGE-2	.927	.976	.770	.794	.855	.842	.879	.903	.721	**.891**	.721	.855	.794	.648	.818	.903
ROUGE-3	.879	.927	.588	.697	.818	.818	.867	.927	.758	.842	.636	.745	.806	.564	.709	.855
ROUGE-4	.818	.879	.721	.697	.745	.745	.867	.867	.685	.794	.564	.612	.830	.455	.709	.758
ROUGE-L	.927	.830	.661	.600	.806	.818	.879	.806	.842	.770	.576	.612	.636	.709	.879	.697
ROUGE-S(∞)	.939	.939	.673	.818	.794	.818	.818	.927	.770	.879	.636	.818	.697	.527	.709	.867
ROUGE-S(9)	.879	.952	.600	.745	.721	.794	.733	.939	.758	.806	.576	.806	.673	.564	.745	.855
ROUGE-S(4)	.891	.964	.600	.794	.794	.794	.794	.939	.709	.842	.576	.770	.770	.733	.758	.842
ROUGE-SU(∞)	.939	.939	.673	.818	.794	.818	.818	.927	.770	.879	.636	.818	.697	.553	.709	.867
ROUGE-SU(9)	.879	.964	.600	.745	.721	.794	.745	.939	.745	.806	.576	.758	.612	.564	.745	.903
ROUGE-SU(4)	.879	**.988**	.600	.745	.721	.770	.794	.903	.758	.855	.576	.794	.709	.612	.794	.842
$F^{d=2}_{\rm esk}(\beta{=}2)$	.952		.879		.855		.939		.842		.927		**.903**		.903	
$F^{d=3}_{\rm esk}(\beta{=}3)$	.952		**.915**		.891		.939		.855		.903		**.903**		.903	
$F^{d=3}_{\rm esk}(\beta{=}2)$	.964		.867		.867		.976		.818		**.927**		.879		.879	
$F^{d=3}_{\rm esk}(\beta{=}3)$	.964		.891		**.915**		.976		.758		.903		.709		.891	
$F^{d=4}_{\rm esk}(\beta{=}2)$	.927		.830		.867		.952		.661		.903		.733		**.915**	
$F^{d=4}_{\rm esk}(\beta{=}3)$	.927		.842		.842		**.988**		.588		.903		.673		.891	
$F^{d=2}_{\rm wsk}(\beta{=}2)$	.976		.794		.830		.952		.818		.867		.806		.891	
$F^{d=2}_{\rm wsk}(\beta{=}3)$	.952		.842		.830		.952		.818		.867		.794		.903	
$F^{d=3}_{\rm wsk}(\beta{=}2)$	.976		.794		.818		.939		.806		.855		.733		.879	
$F^{d=3}_{\rm wsk}(\beta{=}3)$	.976		.879		.855		.952		.806		.818		.794		**.915**	
$F^{d=4}_{\rm wsk}(\beta{=}2)$	.964		.794		.818		.939		.806		.855		.697		**.915**	
$F^{d=4}_{\rm wsk}(\beta{=}3)$	.964		.867		.855		.976		.745		.855		.770		**.915**	

Table 6: Best scores for each data set.

Pearson's Correlation Coefficient

Length	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{\rm avg}$
short	.945	.946	.933	.967
(d, λ, β)	(2,0.7,2)	(2,0.7,4)	(2,0.1,3)	(2,0.7,3)
long	.941	.962	.971	.972
(d, λ, β)	(2,0.6,2)	(2,0.6,3)	(2,0.7,2)	(2,0.8,2)

Spearman's Rank Correlation Coefficient

Length	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{\rm avg}$
short	.964	.915	.915	.988
(d, λ, β)	(3,0.9,4)	(2,0.3,4)	(3,0.5,3)	(4,0.7,4)
long	.855	.927	.915	.939
(d, λ, β)	(2,0.8,4)	(3,0.5,2)	(2,0.5,4)	(2,0.8,3)

for d and length of summary and correlates closely with human evaluation results. Moreover, it includes no trivial way of obtaining a good score. These are significant advantages over ROUGE family. In addition, our method outperformed the WSK-based method in most cases. This result confirms the effectiveness of semantic information and the significant advantage of the ESK.

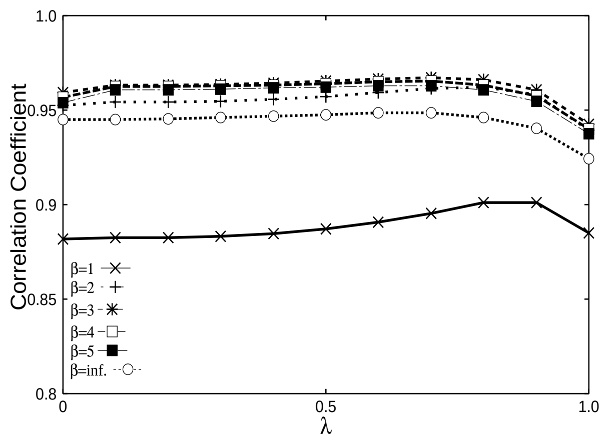
4.6 Effects of Parameters

Our method has three parameters, d, λ, and β. In this section, we discuss the effects of these parameters. Figure 1 shows r and ρ for various λ and β values with respect to $\mathcal{D}_{\rm avg}$. Note that we set d at 2 in the figure because the tendency is similar when we use other values, namely $d(=3 \text{ or } 4)$. From Fig. 1, we can see that $\beta{=}1$ is not good. With automatic summarization, 'precision' is not necessarily a good evaluation measure because highly redundant summaries may obtain a very high precision. On the other hand, 'recall' is not good when a system's output is redundant. Therefore, equal treatment of 'precision' and 'recall' does not give a good evaluation measure. The figure shows that $\beta{=}2, 3$ and 5 are good for r and $\beta{=}3, 4, 5$ and infinity are good for ρ.

Moreover, we can see a significant differences between $\lambda = 1$ and others from the figure. This implies an advantage of our method compared to ROUGE-S and ROUGE-SU, which cannot handle decay factor for skip-n-grams.

From Fig. 1, we can see that ρ is more sensitive to β than r. Here, $\beta{=}3, 4, 5$ and infinity obtained the best results. $\beta{=}1$ was again the worst. This result indicates that we have to determine the parameter value properly for different tasks. λ does not greatly affect the correlation for $d{=}3, 4, 5$ and infinity as regards the middle range.

Table 6 show the best results when we examined all parameter combinations. In the brackets, we show the best settings of these parameter combinations. For r, $d{=}\ 2$ provides the best result and middle range λ and $\beta{=}2$ or 3 are good in most cases. On the other hand, the best settings for ρ vary with



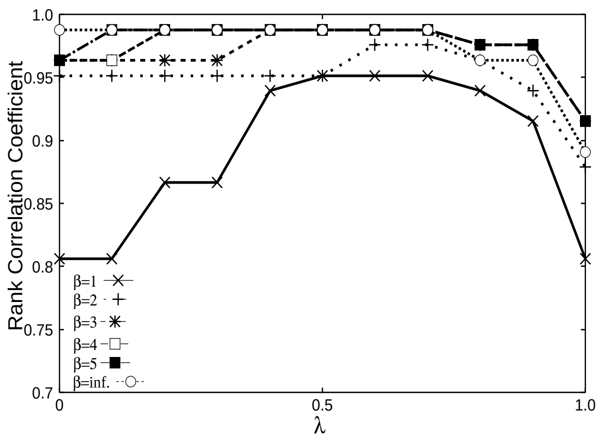


Figure 1: Correlation coefficients for various values of β and λ on $\mathcal{D}_{\text{avg}}$.

the data set. $d=2$ is not always good for ρ.

In short, we can see that the decay parameter for skips is significant and long skip-n-grams are effective especially ρ.

These results show that our method has an advantage over the ROUGE family. In addition, our method is robust and sufficiently good even if close attention is not paid to the parameters.

5 Conclusion

In this paper, we described an automatic evaluation method based on the ESK, which is a method for measuring the similarities between texts based on sequences of words and word senses. Our experiments showed that our method is comparable to ROUGE family for short summaries and outperforms it for long summaries. In order to prove that our method is language independent, we will conduct an experimental evaluation by using DUC's evaluation data. We believe that our method will also be useful for other natural language generation tasks. We are now planning to apply our method to an evaluation of machine translation.

References

N. Cancedda, E. Gaussier, C. Goutte, and J-M. Renders. 2003. Word Sequence Kernels. *Journal of Machine Learning Research*, 3(Feb):1059–1082.

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. In *Proc. of Neural Information Processing Systems (NIPS'2001)*.

D. Harman and P. Over. 2004. The Effects of Human Variation in DUC Summarization Evaluation. In *Proc. of Workshop on Text Summarization Branches Out*, pages 10–17.

T. Hirao, T. Fukusima, M. Okumura, C. Nobata, and H. Nanba. 2004a. Corpus and Evaluation Measures for Multiple Document Summarization with Multiple Sources. In *Proc. of the COLING*, pages 535–541.

T. Hirao, J. Suzuki, H. Isozaki, and E. Maeda. 2004b. Dependency-based Sentence Alignment for Multiple Document Summarization. In *Proc. of the COLING*, pages 446–452.

C. Hori, T. Hori, and S. Furui. 2003. Evaluation Methods for Automatic Speech Summarization. In *Proc. of the Eurospeech2003*, pages 2825–2828.

S. Ikehara, M. Miyazaki, S. Shirai, A. Yokoo, H. Nakaiwa, K. Ogura, Y. Ooyama, and Y. Hayashi. 1997. *Goi-Taikei – A Japanese Lexicon (in Japanese)*. Iwanami Shoten.

C-Y. Lin and E. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proc. of the NAACL/HLT*, pages 150–157.

C-Y. Lin and F.J. Och. 2004. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. In *Proc. of the ACL*, pages 606–613.

C-Y. Lin. 2004a. Looking for a Good Metrics: ROUGE and its Evaluation. In *Proc. of the NTCIR Workshops*, pages 1–8.

C-Y. Lin. 2004b. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proc. of Workshop on Text Summarization Branches Out*, pages 74–81.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text Classification using String Kernel. *Journal of Machine Learning Research*, 2(Feb):419–444.

K. Papineni, S. Roukos, T. Ward, and Zhu W-J. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of the ACL*, pages 311–318.

R. Soricut and E. Brill. 2004. A Unified Framework for Automatic Evaluation using N-gram Co-occurrence Statistics. In *Proc. of the ACL*, pages 614–621.

Discretization Based Learning Approach to Information Retrieval

Dmitri Roussinov
Department of Information Systems
W.P. Carey School of Business
Arizona State University

Tempe, AZ, 85287
dmitri.roussinov@asu.edu

Weiguo Fan
Department of Information Systems and Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
wfan@vt.edu

Abstract

We approached the problem as *learning how to order documents* by estimated relevance with respect to a user query. Our *support vector machines* based classifier learns from the relevance judgments available with the standard test collections and generalizes to *new, previously unseen queries*. For this, we have designed a representation scheme, which is based on the *discrete* representation of the local (*lw*) and global (*gw*) weighting functions, thus is capable of reproducing and enhancing the properties of such popular ranking functions as *tf.idf*, *BM25* or those based on language models. Our tests with the standard test collections have demonstrated the capability of our approach to *achieve the performance of the best known scoring functions* solely from the labeled examples and without taking advantage of knowing those functions or their important properties or parameters.

1. Introduction

Our work is motivated by the objective to bring closer numerous achievements in the domains of machine learning and classification to the classical task of ad-hoc information retrieval (IR), which is ordering documents by the estimated degree of relevance to a given query. Although used with striking success for text categorization, classification-based approaches (e.g. those based on support vector machines, Joachims, 2001) have been relatively abandoned when trying to improve ad hoc retrieval in favor of empirical (e.g. vector space, Salton & McGill, 1983) or generative (e.g. language models; Zhai & Lafferty 2001; Song & Croft; 1999), which produce a ranking function that gives each document a score, rather than trying to learn a classifier that would help to discriminate between relevant and irrelevant documents and order them accordingly. A generative model needs to make assumptions that the query and document words are sampled from the same underlying distributions and that the distributions have certain forms, which entail specific smoothing techniques (e.g. popular Dirichlet-prior). A discriminative (classifier-based) model, on the other side, does not need to make any assumptions about the forms of the underlying distributions or the criteria for the relevance but instead, learns to predict to which class a certain pattern (document) belongs to based on the labeled training examples. Thus, an important advantage of a discriminative approach for the information retrieval task, is its ability to explicitly utilize the relevance judgments existing with standard test collections in order to train the IR algorithms and possibly enhance retrieval accuracy for the new (unseen) queries.

Cohen, Shapire and Singer (1999) noted the differences between ordering and classification and presented a two-stage model to learn ordering. The first stage learns a classifier for preference relations between objects using any suitable learning mechanism (e.g. support vector machines; Vapnik, 1998). The second stage converts preference relations into a rank order. Although the conversion may be NP complete in a general case, they presented efficient approximations. We limited our first study reported here to linear classifiers, in which conversion can be performed by simple ordering according to the score of each document. However, approaching the problem as "learning how to order things" allowed us to design our sampling and training mechanisms in a novel and, we believe, more powerful way.

Our classifier learns how to compare every pair of documents with respect to a given query, based on the relevance indicating features that the documents may have. As it is commonly done in information retrieval, the features are derived from the word overlap between the query and documents. According to Nallapati (2004), the earliest formulation of the classic IR problem as a classification (discrimination) problem was suggested by Robertson and Sparck Jones (1976), however performed well only when the relevance judgments were available for the same query but not generalizing well to new queries. Fuhr and Buckley (1991) used polynomial regression to estimate the coefficients in a linear scoring function combining such well-known features as a weighted term frequency, document length and query length. They tested their "description-oriented" approach on the standard small-scale collections (Cranfield, NPL, INSPEC, CISI, CACM) to achieve the relative change in the average precision ranging from -17%

to + 33% depending on the collection tested and the implementation parameters. Gey (1994) applied logistic regression in a similar setting with the following results: Cranfield +12%, CACM +7.9%, CISI -4.4%, however he did not test them on new (unseen by the algorithm) queries, hypothesizing that splitting documents into training and testing collections would not be possible since “a large number of queries is necessary in order to train for a decent logistic regression approach to document retrieval.” Instead, he applied a regression trained on Cranfield to CISI collection but with a negative effect.

Recently, the approaches based on learning have reported several important breakthroughs. Fan et al. (2004) applied genetic programming in order to learn how to combine various terms into the optimal ranking function that outperformed the popular Okapi formula on robust retrieval test collection. Nallapati (2004) made a strong argument in favor of discriminative models and trained an SVM-based classifier to combine 6 different components (terms) from the popular ranking functions (such as tf.idf and language models) to achieve better than the language model performance in 2 out of 16 test cases (figure 3 in Nallapati, 2004), not statistically distinguishable in 8 cases and only 80% of the best performance in 6 cases. There have been studies using past relevance judgements to optimize retrieval. For example, Joachims (2002) applied Support Vector Machines to learn linear ranking function from user click-throughs while interfacing with a search engine.

In this study, we have developed a representation scheme, which is based on the discretization of the *global* (corpus statistics) and *local* (document statistics) weighting of term overlaps between queries and documents. We have empirically shown that this representation is flexible enough to learn the properties of the popular ranking functions: tf.idf, BM25 and the language models. The major difference of our work from Fan et al. (2004) or Nallapati (2004) or works on fusion (e.g. Vogt & Cottrell, 1999) is that *we did not try to combine several known ranking functions (or their separate terms) into one, but rather we learn the weighting functions directly through discretization.*

Discretization allows representing a continuous function by a set of values at certain points. These values are learned by a machine learning technique to optimize certain criteria, e.g. average precision.

Another important motivation behind using *discretization* was to design a representation with high dimensionality of features in order to combine our representation scheme with Support Vector Machines (SVM) (Vapnik, 1998), which are known to work well with a large number of features. SVM contains a large class of neural nets, radial margin separation (RBF) nets, and polynomial classifiers as special cases. They have been delivering superior performance in classification tasks in general domains, e.g. in face recognition (Hearst, 1998), and in text categorization (Joachims, 2001).

Another important distinction of this work from the prior research is that we train our classifier not to predict the absolute relevance of a document *d* with respect to a query *q*, but rather to predict which of the two documents *d1, d2* is more relevant to the query *q*. The motivation for this distinction was that all the popular evaluation metrics in information retrieval (e.g. average precision) are based on document ranking rather than classification accuracy. This affected our specially designed sampling procedure which we empirically discovered to be crucial for successful learning.

We have also empirically established that our combination of the representation scheme, learning mechanism and sampling allows learning from the past relevance judgments in order to successfully generalize to the new (unseen) queries. When the representation was created without any knowledge of the top ranking functions and their parameters, *our approach reached the known top performance solely through the learning process*. When our representation was taking advantage of functions that are known to perform well and their parameters, the resulting combination was *able to slightly exceed the top performance on large test collections*. The next section formalizes our Discretization Based Learning (DBL) approach to Information Retrieval, followed by empirical results and conclusions.

2. Formalization Of Our Approach

2.1 Query and Document Representation

We limit our representation to the so called *lw.gw* *class*: $R(q, d) = \sum_{t \subset q} L(tf(t,d), d)G(t)$,

where *L*, local weighting, is the function of the number of occurrences of the term in the document *tf*, possibly combined with the other document statistics, e.g. word length. *G(t)*, global weighting, can be any collection level statistic of the term. For example, in the classical *tf.idf* formula *L(tf, d) = tf / |d|*, where *tf* is the number of occurrences of the term *t* in the document, |*d*| is the length of the document vector and *G(t) = log (N / df(t))*, where *df(t)* is the total number of documents in the collection that have term *t* and *N* is the total number of documents.

Without loss of generality it may also be extended to handle a number of occurrences of the term in the query, but we omit it here in our formalization for simplicity. *Lw.gw* class includes the BM25 Okapi ranking function which performs well on TREC collections (Robertson et al., 1996). It can be shown that many of the recently introduced language models fall into that category as well, specifically the best performing in TREC ad hoc tests Dirichlet smoothing, Jelinek Mercer smoothing, and Absolute Discounting approaches can be represented that way (see equation 6 and table I in Zhai & Lafferty, 2001). An *lw.gw* representation of Jelinek Mercer smoothing was used in Nallapati (2004). It has been known for a long time that the shapes of the global and local weighting functions can *dramatically affect the precision* in standard test collections because it in fact determines the difference between such formulas as *tf.idf*, *bm25* and language models. *However, we are not aware of any attempts to learn those shapes directly from the labeled examples, which we performed in this study.*

2.2 Intuition behind the discretization-based learning

The intuition behind discretization approach is to represent a function by values at the finite number of points. Then, the optimal shape of the function can be learned by using one of the machine learning techniques. Our discretization based learning (DBL) approach to information retrieval learns how important each class of an occurrence of a query term in a document. For example, in some very "primitive" DBL approach, we can define two classes: Class S ("strong"), containing all multiple occurrences of a rare query term (e.g. "discretization") in a document and Class W ("weak"), containing all single occurrences of a frequent term (e.g. "information"). Then, the machine learning technique should discover that the occurrences of Class S are much stronger indicators of relevance than the occurrences of Class W. In the DBL implementation presented in this paper, each occurrence of a query term is assigned to a class (called *bin*) based on the term document frequency in the collection (*df*) and the number of occurrences within the document (*tf*). The bin determines the weight of the contribution of each occurrence of the query term in the ranking score. Thus, the relevance score is just the weighted sum of the numbers of occurrences within each bin. The other way of looking at it is that the score is produced by a linear classifier, where the total number of occurrences within each bin serves as the feature value. *By learning the optimal weights, a linear classifier effectively learns the optimal shapes of the global (gw) and local (lw) weighting functions. By learning the discrimination properties of each bin, rather than separate word terms, DBL method allows generalization to new queries.*

2.3 Discretizing global weighting

We discretized the shape of the G*(t)* function by assigning each term to its global weighting bin *g*, which is an integer number in the *[1, |B|]* range, |*B*| is the total number of global weighting bins. The assignment of the term *t* to its global weighting bin *g(t)* is performed on the log linear scale according to the document frequency *df* of the term:

$$g(t) = \{|B|(1 - \frac{\log(df(t))}{\log(N)})\} \qquad (1)$$

where *N* is the total number of documents, {.} stands for rounding down to the nearest integer. The logarithmic scale allows more even term distribution among bins than simple linear assignment, which is desirable for more efficient learning. It is motivated by a typical histogram of *df(t)* distribution, which looks much more uniform in a logarithmic scale. It is important to note that it does not have anything to do with the *log* function in the classical *idf* weighting and that the formula for *g(t)* does not produce any weights but only assigns each term occurrence to a specific bin based on the term document frequency. The weights are later trained and effectively define any shape of global weighting, including such simple functions tried in the prior heuristic explorations as logarithm, square root, reciprocal and others.

2.4 Discretizing local weighting

Similarly to the global weighting, we assigned each occurrence of a term to its local weighting bin *l*, but this time by simply capping *tf* at the total number of local weighting bins |*L*|:

$$l\,(tf(t,d), d) = min(\,tf\,(t,d), |L|)\,) \qquad (1a)$$

Let's note that this particular representation does not really need rounding since *tf* is already a positive integer. However, in a more general case, *tf* can be normalized by document length (as is done in BM25 and language models) and thus local weighting would become a continuous function. It is important to note that our discrete representation does not ignore the occurrences above |*L*| but simply treats them the same way as *tf* = |*L*|. The intuition behind capping is that increasing *tf* above certain value (|L|) would not typically indicate the higher relevance of the document. Typically, a certain number of occurrences is enough to indicate the presence of the relevant passage. Please note again that this bin assignment does not assign any heuristic weights to the term occurrences.

2.5 Final discretized ranking function

The bin assignments based on *tf* and *df* specified in sections 2.3 and 2.4 are straightforward and do not involve any significant "feature engineering." Each occurrence of a query term in a document corresponds to a local/global bin combination *(g, l)*. Each *(g,l)* combination determines a feature in a vector representing a document-query pair ***f****(d, q)* and is denoted below as *f(d, q) [g , l]* . The dimensionality of the feature space is |*L*| x |*B*|. E.g. for 8 local weighting bins and 10 global weighting bins we would deal with the vector size of 80. A feature vector ***f****(d, q)* represents each document *d* with respect to query *q*. The value of each feature in the vector is just the number of the term occurrences assigned to the pair of bins *(g, l):*

$$f\,(d,q)\,[g\,,l] \;=\; \sum_{t \subset q,\, g(t)=g,\, l(t,d)=l} 1 \qquad (2)$$

Since our features capture local (tf) and global (df) term occurrence information, in order to represent a ranking function, we can simply use the dot product between the feature vector and the vector of learned optimal weights **w**:

R(q, d) = **w** * ***f*** *(d, q).*

Ideally, the learning mechanism should assign higher weights to the more important bin combinations (e.g. multiple occurrence of a rare term) and low weights to the less important combinations (e.g. single occurrence of a common term). The exact learned values determine the optimal shape of global and local weighting.

We still can make the representation more powerful by considering the learned weights *w[g, l]* not the replacements but rather the adjustments to some other chosen global *G (t)* and local *L (t, d)* weighting functions:

$$f(d, q)[g, l] = \sum_{t \subset q,\, g(t)=g,\, l(tf(t,d),d)=l} L(t,d)G(t) \qquad (2a)$$

We define the specific choice of global *G()* and local *L()* weighting functions as *starting ranking function (SRF)*. When all the bin weights *w[g, l]* are set to *1*, our ranking function is the same as its *SRF*. The learning process finds the optimal values for *w[g, l]* for the collection of training queries and their relevance judgments, thus adjusting the important shapes of the global and local weighting to achieve better accuracy. *SRF* can be chosen from one of the known to perform well ranking functions (e.g. *tf.idf* or *BM25* or based on language models) to take advantage of the fact that those formulas and their optimal parameters on the standard test collections are known for the researchers. Alternatively, we can set *SRF* to the constant value (e.g. *1* in formula 2), thus not taking advantage of any of the prior empirical investigations and to see if our framework is able to learn reasonable (or even top-notch) performance purely from labeled examples. Below, we describe our experiments with each approach.

Since the score is linear with respect to the feature values, we can train the weights **w** as a linear classifier that predicts the preference relation between pairs of documents with respect to the given query. Document *d1* is more likely to be relevant (has a higher score) than document *d2* iff ***f****(d1, q)* * ***w*** > ***f****(d1, q)* * ***w***. An important advantage of using a linear classifier is that rank ordering of documents according to the learned pairwise preferences can be simply performed by ordering according to the linear score. Please refer to Cohen et al. (1999) for the ordering algorithms in a more general non linear case.

We chose support vector machines (SVM) for training the classifier weights *w[g, l]* since they are known to work well with large numbers of features, ranging in our experiments from 8 to 512, depending on the number of bins. For our empirical tests, we used the SVMLight package freely available for academic research from Joachims (2001). We preserved the default parameters coming with version V6.01. Although SVMLight package allows learning ranking, we opted for training it as a classifier to retain more control over sampling, which we found crucial for successful learning, as described in the section below.

2.6 Sampling

Since we were training a classifier to predict preference relations, but not the absolute value of relevance, we trained on the differences between feature vectors. Thus, for each selected (sampled) pair of documents (*dr, di*), such that *dr* is a relevant document and *di* is irrelevant, the classifier was presented with a positive example created from the vector of differences of features $\boldsymbol{f_p} = \boldsymbol{f}(q, dr) - \boldsymbol{f}(q, di)$, and also with the negative example as the inverse of it: $\boldsymbol{f_n} = \boldsymbol{f}(q, di) - \boldsymbol{f}(q, dr)$. This approach also balances positive and negative examples.

We also informally experimented with training on absolute relevance judgments, similar to the prior work mentioned in the Introduction but obtained much worse results. We explain it by the fact that relative judgments (pairwise comparisons) are more generalizable to new queries than absolute judgments (relevant/irrelevant). This may explain prior difficulties with applying discriminative approaches mentioned in our Introduction.

Since presenting all pairs to the training mechanism would be overwhelming, we performed pseudo-random sampling of documents by the following intuitive consideration. Since it is more efficient to present the classifier with the pairs from the documents that are likely to more strongly affect the performance metric (average precision), we first pre-ordered the retrieved documents by any of the reasonably well-performing scoring function (e.g. *tf.idf*) and limited the sample of documents to the top *1000*. Then, for each query, each known relevant document *dr* from that subset was selected and "paired" with a certain number of randomly selected irrelevant documents. This number was linearly decreasing with the position of the relevant document in the pre-order. Thus, the higher the document was positioned in the pre-order, the more times it was selected for pairing (training). This placed more emphasis at correctly classifying the more important document pairs in the average precision computation. Again, without the correct emphasis during sampling the obtained results were much weaker. However, the choice of the ranking function to perform pre-order was found to be not important: virtually the same results were obtained using *tf.idf* or bm25 or language models.

3. Empirical Evaluation

3.1 Empirical setup

We used the TREC, Disks 1 and 2, collections to test our framework. We used topics 101-150 for training and 151-200 for testing and vice-versa. For indexing, we used the Lemur package (Kraaij et al., 2003), with the default set of parameters, and no stop word removal or stemming. Although those procedures are generally beneficial for accuracy, it is also known that they do not significantly interfere with testing various ranking functions and thus are omitted in many studies to allow easier replication.

We used only topic titles for queries, as it is commonly done in experiments, e.g. in Nallapati (2004). We used the most popular average (non-interpolated) precision as our performance metric, computed by the script included with the Lemur toolkit (later verified by trec_eval). The characteristics of the collection after indexing are shown in Table 1. We also reproduced results similar to the reported below on the Disk 3 collection and topics 101-150, but did not include them in this paper due to size limitations.

Collection	TREC Disks 1 and 2
Number of documents	741,863
Number of terms	325,059,876
Number of unique terms	697,610
Average doc. length	438
Topics	101-200

Table 1. The characteristics of the test collection: TREC Disks 1,2

3.2 The baseline

In this study, we were interested exclusively in the improvements due to learning, thus still staying within the "bag of words" paradigm. Although many enhancements can be easily combined within our framework, we limited our search for the baseline performance to "bag of words" techniques to avoid unfair comparison. We used the results reported in Nallapati (2004) as guidance and verified that the best performing language model on this test collection was the one based on the Dirichlet smoothing with $\mu = 1900$. Our average precision was lower (*0.205* vs. *0.256*), most likely due to the different indexing parameters, stemming or using a different stopword list. By experimenting with the other ranking functions and their parameters, we noticed that the implementation of *BM25,* available in Lemur, provided almost identical performance (*0.204*). Its ranking function is $BM25(tf, df) = tf / (tf + K^* (1 - b + b * |d| / |d|_a) * \log (N / (df + .5))$, where $|d|$ is the document word length and $|d|_a$ is its average across all documents. The optimal parameter values were close to the default $K = 1.0$ and $b = .5$. We noticed that the query term frequency components could be ignored without any noticeable loss of precision. This may be because the TREC topic titles are short and the words are very rarely repeated in the queries. Since the difference between this ranking function and the optimal from the available language models was negligible we selected the former as both our baseline and also as the starting ranking function (SRF) in our experiments. For simplicity, we call it simply *BM25* throughout our paper.

3.3 Discretization accuracy

Before testing the learning mechanism, we verified that the loss due to discretization is minimal and thus the approach is capable of capturing global and local weighting. For this, we discretized our baseline BM25 formula replacing each score contribution of the occurrence of a term $G(t)L(t,d) = BM25(t, d)$ with its average across all other occurrences within the same bin combination *[g, l]*, which is determined by the formulas 1 and 1a. We discovered that for the $|B| \times |L| = 8 \times 8$ configuration, the loss in average precision did not exceed *2%* (relatively). This demonstrates that the *G(t)L(t,d)* ranking functions can be discretized (replaced by values at certain points) at this level of granularity without losing much accuracy. We also verified that the weights *w[g, l]* can affect the performance significantly: when we set them to random numbers in the [0,1] range, the performance dropped by 50% relatively to the baseline.

3.4 Ability to achieve top performance from scratch

First, we were curious to see if our framework can learn reasonable performance without taking advantage of our knowledge of the top ranking functions and their parameters. For this, we set our starting ranking function (SRF) to a constant value, thus using only the minimum out of the empirical knowledge and theoretical models developed by information retrieval researchers during several decades: specifically only the fact that relevance can be predicted by tf and df

Table 2 shows performance for the 16 x 8 combination of bins. It can be seen that our approach has reached 90-100% of the top performance (baseline) solely through the learning process. The *original* performance is the one obtained by assigning all the classifier weights to 1. It can be seen that the topics 151-200 are more amenable for the technique that is why they show better recovery when used as a test set even when the training set 101-150 recovers only 90%. In order to evaluate if more training data can help, we also ran tests using 90 topics for training and the remaining 10 for testing. We ran 10 tests each time using 10 different sequential topics for testing and averaged our results. In this case, *the averaged performance was completely restored to the baseline level* with the mean difference in precision across test queries +0.5% and 1% standard deviation of the mean.

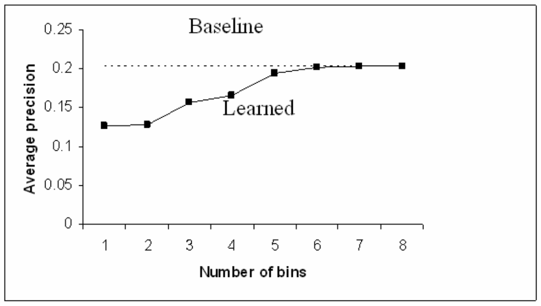


Figure 1. Learning local weighting for various

Testing:	**101-150**			**151-200**		
Training:	Original	Learned	Baseline	Original	Learned	Baseline
101-150	.119	.165	.174	.135	.180	.204
151-200	.119	.175	.174	.135	.206	.204

Table 2. Learning without any knowledge of ranking functions. 16 x 8 bin design.

Testing:	101-150			151-200		
Training:	Learned	Baseline	% change	Learned	Baseline	
101-150	.180	.174	+2.3 (+/- 0.9)	.208	.204	+2.3 (+/- 1.0)
151-200	.179	.174	+1.8 (+/- 1.0)	.210	.204	+3.2 (+/- 1.3)

Table 3. Surpassing the baseline performance. 8 x 8 bin design.

numbers of bins. Learning on 101-150 and testing on 151-200.

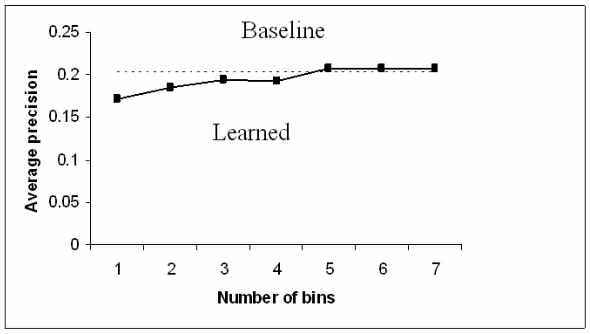


Figure 2. Learning global weighting for various numbers of bins. Learning on 101-150 and testing on 151-200.

We believe this is a remarkable result considering the difficulties that the prior learning based approaches had with the classical information retrieval task. We attribute our success to both higher flexibility and generalizability of our discrete representation. We also varied the number of bins to evaluate the effect of granularity of representation. Figures 1 and 2 demonstrate that 8 bins suffice for both global and local weighting. Higher numbers did not result in noticeable improvements.

When the same set was used for training and testing the result obviously overestimates the learning capability of the framework. However, it also gives the upper bound of performance of a discretized *gw.lw* combination assuming that the loss due to discretization is negligible which can be easily attained by using sufficiently large number of bins. *Thus, the results indicate that gw.lw, which includes practically all the popular "bag of words" ranking formulas such as tf.idf, BM25 or language models, has almost reached its upper limit and other classes of representations and ranking formulas need to be explored to attempt greater improvements.*

Figure 2. Learning global weighting for various numbers of bins. Learning on 101-150 and testing on 151-200.

3.5 Ability to surpass top performance

In order to test whether our approach can exceed the baseline performance we set *BM25* to be our starting ranking function (SRF). Thus, in this case:

$G(t) = \log (N / (df + .5))$ (6)

$L(tf, d) = tf / (tf + K * (1 - b + b * |d| / |d|_a)$

Table 3 shows performance for the 8 by 8 bin design. Although the improvement is relatively small (2-3%) it is still statistically significant at the level of alpha < 0.1, when the paired t-test was performed. The value in "*% change*" column shows the mean % improvement across all the queries and its standard deviation. It may differ from the % change of the mean performance since there is wide variability in the performance across queries but smaller variability in the improvement.

We believe even such a small improvement is remarkable considering the amount of attention the researches have paid to optimizing the ranking functions for this specific data set which has been available for more than seven years. A number of recent studies reported comparable improvements on the same test collection by using more elaborate modeling or richer representations. Of course the improvement due to the techniques such as those based on n-grams, document structures, natural language processing or query expansion can possibly achieve even better results. However in this study we deliberately limited our focus to the "bags of words."

3.6 Shape of optimal local weighting

Figure 3 shows the optimal shape of the local weighting function *L(tf)* learned on entire set of 100 topics and plotted against their counterparts of *BM25(t, d) = tf / (tf + 1)* and *tf.idf(t, d) = tf* for comparison. For plotting purposes, we assumed that the document length was equal to its average. The values were linearly scaled to meet at the *tf = 8* point. It is easy to observe that the behavior of the optimal function is much closer to BM25 than to *tf.idf*, which explains the good performance of the former on this test set.

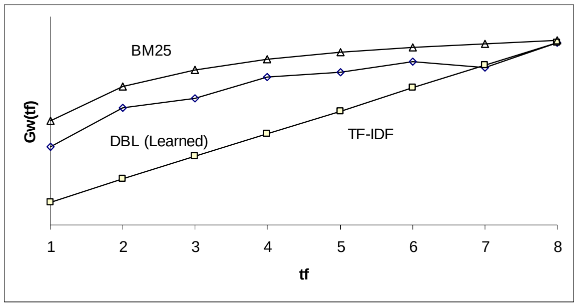


Figure 3. Learned optimal shape of local weighting.

3.7 Shape of optimal global weighting

Figure 4 shows the optimal shape of the global weighting function *G(t)* learned on the entire set of 100 topics with $|B| = 32$ plotted in logarithmic scale against the popular *idf* weighting used in both *tf.idf* and BM25. The lower end of the X-axis (*log10 df < 2*) corresponds to very infrequent terms, so the learned weights may not be very informative since the classifier encounters fewer occurrences of them and their impact on the overall accuracy is small. In the mid range (5,000 – 10,000), the optimal weights are higher than idf, which indicates that the latter has an overly steep shape to discount high frequency terms. A more detailed interpretation of the optimal shape may require further investigation.

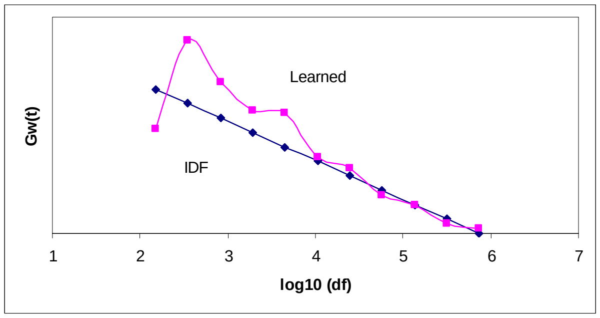


Figure 4. Learned optimal shape of global weighting G(t).

4. Conclusions

We explored learning how to rank documents with respect to a given query using linear Support Vector Machines and discretization-based representation. Our approach represents a family of discriminative approaches, currently studied much less than heuristic (*tf.idf*, *bm25*) or generative approaches (language models). Our experiments indicate that *learning from relevant judgments available with the standard test collections and generalizing to new queries is not only feasible but can be a source of improvement.* When tested with a popular standard collection, our approach achieved the performance of the best well-known techniques (BM25 and language models), which have been developed as a result of extensive past experiments and elaborate theoretical modeling. When combined with the best performing ranking functions, our approach added a small (2-3%), but statistically significant, improvement.

Although practical significance of this study may be limited at the moment since it does not demonstrate a dramatic increase in retrieval performance in large test collections, we believe our findings have important theoretical contributions since they indicate that the power of discriminative approach is comparable to the best known analytical or heuristic apporaches. This work also lays the foundation for extending the discriminative approach to "richer" representations, such as those using word n-grams, grammatical relations between words, and the structure of documents.

Our results also indicate that *gw.lw* family, which includes practically all the popular "bag of words" ranking formulas such as tf.idf, BM25 or language models, has almost reached its upper limit and other classes of representations and ranking formulas need to be explored in order to accomplish significant performance break-troughs.

Of course, using only few test cases (topics sets and collections) is a limitation of this current study, which we are going to address in our future research. We view our approach as a complement, rather than competitive, to the analytical approaches such as language models. Our approach can be also used as an explorative tool in order to identify important relevance-indicating features, which can be later modeled analytically. We believe that our work and the ones referred in this paper may bring many of the achievements made in a more general area of classification and machine learning closer to the task of rank ordered information retrieval, thus making retrieval engines more helpful in reducing the information overload and meeting people's needs.

5. Acknowledgement

Weiguo Fan's work is supported by NSF under the grant number ITR0325579.

References

Bartell, B., Cottrell, G., and Belew, R.(1994). Optimizing Parameters in a Ranked Retrieval System Using Multi-Query Relevance Feedback. *Symposium on Document Analysis and Information Retrieval (SDAIR).*

Chengxiang Zhai and John Lafferty (2001). A study of smoothing methods for language models applied to Ad Hoc information retrieval. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 334 – 342, 2001.

Cohen, W., Shapire, R., and Singer, Y. (1999). Learning to order things. Journal of Artificial Intelligence Research, 10, 243-270, 1999.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 194--202). Tahoe City, CA: Morgan Kaufmann.

Fan, W., Luo, M., Wang, L., Xi, W., and Fox, A. (2004). Tuning Before Feedback: Combining Ranking Discovery and Blind Feedback for Robust Retrieval. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

Fuhr, N. and C. Buckley (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems, 9*, 223–248.

Fuhr, N. and C. Buckley (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems, 9*, 223–248.

Gey, F. C. (1994). Inferring probability of relevance using the method of logistic regression. In *Proceedings of the 17th ACM*

Conference on Research and Development in Information Retrieval (SIGIR'94), pp. 222–231.

Hearst, M. (1998). Support Vector Machines. *IEEE Intelligent Systems Magazine, Trends and Controversies*, Marti Hearst, ed., 13(4), July/August 1998.

Hun-Nan Hsu, Hung-Ju Huang and Tzu-Tsung Wong (2000). Why Discretization Works for Naive Bayesian Classifiers, In Proceedings of the 17th International Conference on Machine Learning (ICML-2000), Stanford, CA, USA. Page 399-406.

Joachims, T. (2001). A Statistical Learning Model of Text Classification with Support Vector Machines. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

Joachims, T. (2002). Optimizing Search Engines Using Clickthrough Data, *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2002.

Kraaij, W., Westerveld T. and Hiemstra, D. (2003)., The Lemur Toolkit for Language Modeling and Information Retrieval, http://www-2.cs.cmu.edu/~lemur

Nallapati, R. (2004). Discriminative models for information retrieval. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, 2004, pp. 64-71.

Robertson S. E. and Sparck Jones, K. (1976). Relevance weighting of search terms, *Journal of American Society for Information Sciences*, 27(3), pp. 129-146, 1976.

Robertson, S. E., Walker, S., Jones S., Hancock-Beaulieu M.M., and Gatford, M. (1996)., Okapi at TREC-4, in D. K. Harman, editor, Proceedings of the Fourth Text Retrieval Conference, pp. 73–97. NIST Special Publication 500-236, 1996.

Salton, G. and McGill, M.J. (1983). Introduction to Modern Information Retrieval. New York. McGraw-Hill.

Song, F. and W.B. Croft. (1999) A general language model for information retrieval. *In Proceedings of Eighth International Conference on Information and Knowledge Management* (CIKM'99).

Vapnik, V. N. (1998).. Statistical Learning Theory. John Wiley and Sons Inc., New York, 1998.

Vogt, C., G. Cottrell, G. (1999). Fusion Via a Linear Combination of Scores. *Information Retrieval*, 1(3), pp. 151—173.

Local Phrase Reordering Models for Statistical Machine Translation

Shankar Kumar, William Byrne[*]
Center for Language and Speech Processing, Johns Hopkins University,
3400 North Charles Street, Baltimore, MD 21218, U.S.A.
Machine Intelligence Lab, Cambridge University Engineering Department,
Trumpington Street, Cambridge CB2 1PZ, U.K.
skumar@jhu.edu , wjb31@cam.ac.uk

Abstract

We describe stochastic models of local phrase movement that can be incorporated into a Statistical Machine Translation (SMT) system. These models provide properly formulated, non-deficient, probability distributions over reordered phrase sequences. They are implemented by Weighted Finite State Transducers. We describe EM-style parameter re-estimation procedures based on phrase alignment under the complete translation model incorporating reordering. Our experiments show that the reordering model yields substantial improvements in translation performance on Arabic-to-English and Chinese-to-English MT tasks. We also show that the procedure scales as the bitext size is increased.

1 Introduction

Word and Phrase Reordering is a crucial component of Statistical Machine Translation (SMT) systems. However allowing reordering in translation is computationally expensive and in some cases even provably NP-complete (Knight, 1999). Therefore any translation scheme that incorporates reordering must necessarily balance model complexity against the ability to realize the model without approximation. In this paper our goal is to formulate models of local phrase reordering in such a way that they can be embedded inside a generative phrase-based model of translation (Kumar et al., 2005). Although this model of reordering is somewhat limited and cannot capture all possible phrase movement, it forms a proper parameterized probability distribution over reorderings of phrase sequences. We show that with this model it is possible to perform Maximum A Posteriori (MAP) decoding (with pruning) and Expectation Maximization (EM) style re-estimation of model parameters over large bitext collections.

We now discuss prior work on word and phrase reordering in translation. We focus on SMT systems that do not require phrases to form syntactic constituents.

The IBM translation models (Brown et al., 1993) describe word reordering via a distortion model defined over word positions within sentence pairs. The Alignment Template Model (Och et al., 1999) uses phrases rather than words as the basis for translation, and defines movement at the level of phrases. Phrase reordering is modeled as a first order Markov process with a single parameter that controls the degree of movement.

Our current work is inspired by the block (phrase-pair) orientation model introduced by Tillmann (2004) in which reordering allows neighboring blocks to swap. This is described as a sequence of orientations (left, right, neutral) relative to the monotone block order. Model parameters are block-specific and estimated over word aligned trained bitext using simple heuristics.

Other researchers (Vogel, 2003; Zens and Ney, 2003; Zens et al., 2004) have reported performance gains in translation by allowing deviations from monotone word and phrase order. In these cases,

[*] This work was supported by an ONR MURI Grant N00014-01-1-0685.

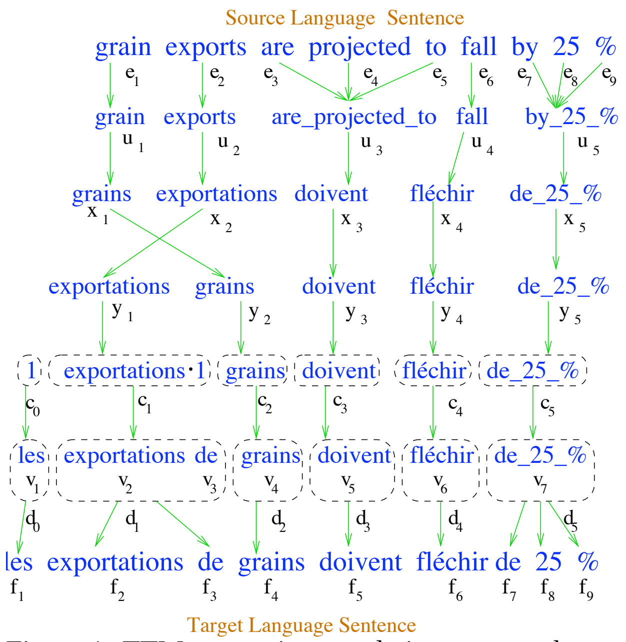

Source Language Sentence
grain exports are projected to fall by 25 %
grain exports are_projected_to fall by_25_%
grains exportations doivent fléchir de_25_%
exportations grains doivent fléchir de_25_%
1 exportations 1 grains doivent fléchir de_25_%
les exportations de grains doivent fléchir de_25_%
les exportations de grains doivent fléchir de 25 %
Target Language Sentence

Figure 1: TTM generative translation process; here, $I = 9, K = 5, R = 7, J = 9$.

reordering is not governed by an explicit probabilistic model over reordered phrases; a language model is employed to select the translation hypothesis. We also note the prior work of Wu (1996), closely related to Tillmann's model.

2 The WFST Reordering Model

The Translation Template Model (TTM) is a generative model of phrase-based translation (Brown et al., 1993). Bitext is described via a stochastic process that generates source (English) sentences and transforms them into target (French) sentences (Fig 1 and Eqn 1).

$$\begin{aligned}
&P(f_1^J, v_1^R, d_0^K, c_0^K, y_1^K, x_1^K, u_1^K, K, e_1^I) = \\
&\quad P(e_1^I)\cdot \\
&\qquad \textit{Source Language Model} \;\; G \\
&\quad P(u_1^K, K|e_1^I)\cdot \\
&\qquad \textit{Source Phrase Segmentation} \;\; W \\
&\quad P(x_1^K|u_1^K, K, e_1^I)\cdot \\
&\qquad \textit{Phrase Translation and Reordering} \;\; R \\
&\quad P(v_1^R, d_0^K, c_0^K, y_1^K|x_1^K, u_1^K, K, e_1^I)\cdot \\
&\qquad \textit{Target Phrase Insertion} \;\; \Phi \\
&\quad P(f_1^J|v_1^R, d_0^K, c_0^K, y_1^K, x_1^K, u_1^K, K, e_1^I) \\
&\qquad \textit{Target Phrase Segmentation} \;\; \Omega
\end{aligned} \tag{1}$$

The TTM relies on a Phrase-Pair Inventory (PPI) consisting of target language phrases and their source language translations. Translation is modeled via component distributions realized as WFSTs (Fig 1 and Eqn 1) : Source Language Model (G), Source Phrase Segmentation (W), Phrase Translation and Reordering (R), Target Phrase Insertion (Φ), and Target Phrase Segmentation (Ω) (Kumar et al., 2005).

TTM Reordering Previously, the TTM was formulated with reordering prior to translation; here, we perform reordering of phrase sequences following translation. Reordering prior to translation was found to be memory intensive and unwieldy (Kumar et al., 2005). In contrast, we will show that the current model can be used for both phrase alignment and translation.

2.1 The Phrase Reordering Model

We now describe two WFSTs that allow local reordering within phrase sequences. The simplest allows swapping of adjacent phrases. The second allows phrase movement within a three phrase window. Our formulation ensures that the overall model provides a proper parameterized probability distribution over reordered phrase sequences; we emphasize that the resulting distribution is not degenerate.

Phrase reordering (Fig 2) takes as its input a French phrase sequence in English phrase order $x_1, x_2, ..., x_K$. This is then reordered into French phrase order $y_1, y_2, ..., y_K$. Note that words within phrases are not affected.

We make the following conditional independence assumption:

$$P(y_1^K|x_1^K, u_1^K, K, e_1^I) = P(y_1^K|x_1^K, u_1^K). \tag{2}$$

Given an input phrase sequence x_1^K we now associate a unique *jump sequence* b_1^K with each permissible output phrase sequence y_1^K. The jump b_k measures the displacement of the k^{th} phrase x_k, i.e.

$$x_k \rightarrow y_{k+b_k}, k \in \{1, 2, ..., K\}. \tag{3}$$

The jump sequence b_1^K is constructed such that y_1^K is a permutation of x_1^K. This is enforced by constructing all models so that $\sum_{k=1}^K b_k = 0$.

We now redefine the model in terms of the jump sequence

$$\begin{aligned}
&P(y_1^K|x_1^K, u_1^K) \\
&= \begin{cases} P(b_1^K|x_1^K, u_1^K) & y_{k+b_k} = x_k \; \forall k \\ 0 & \text{otherwise}, \end{cases}
\end{aligned} \tag{4}$$

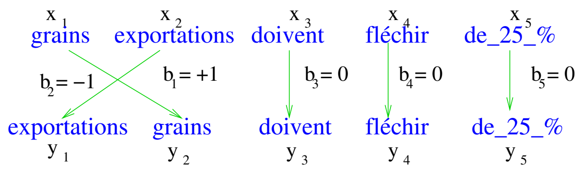


Figure 2: Phrase reordering and jump sequence.

where y_1^K is determined by x_1^K and b_1^K.

Each jump b_k depends on the phrase-pair (x_k, u_k) and preceding jumps b_1^{k-1}

$$P(b_1^K|x_1^K, u_1^K) = \prod_{k=1}^{K} P(b_k|x_k, u_k, \phi_{k-1}), \quad (5)$$

where ϕ_{k-1} is an equivalence classification (state) of the jump sequence b_1^{k-1}.

The jump sequence b_1^K can be described by a deterministic finite state machine. $\phi(b_1^{k-1})$ is the state arrived at by b_1^{k-1}; we will use ϕ_{k-1} to denote $\phi(b_1^{k-1})$.

We will investigate phrase reordering by restricting the maximum allowable jump to 1 phrase and to 2 phrases; we will refer to these reordering models as MJ-1 and MJ-2. In the first case, $b_k \in \{0, +1, -1\}$ while in the second case, $b_k \in \{0, +1, -1, +2, -2\}$.

2.2 Reordering WFST for MJ-1

We first present the Finite State Machine of the phrase reordering process (Fig 3) which has two equivalence classes (FSM states) for any given history b_1^{k-1}; $\phi(b_1^{k-1}) \in \{1, 2\}$. A jump of $+1$ has to be followed by a jump of -1, and 1 is the start and end state; this ensures $\sum_{k=1}^{K} b_k = 0$.

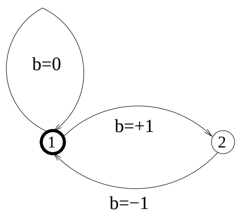


Figure 3: Phrase reordering process for MJ-1.

Under this restriction, the probability of the jump b_k (Eqn 5) can be simplified as

$$P(b_k|x_k, u_k, \phi(b_1^{k-1})) = \quad (6)$$
$$\begin{cases} \beta_1(x_k, u_k) & b_k = +1, \phi_{k-1} = 1 \\ 1 - \beta_1(x_k, u_k) & b_k = 0, \phi_{k-1} = 1 \\ 1 & b_k = -1, \phi_{k-1} = 2. \end{cases}$$

There is a single parameter jump probability $\beta_1(x, u) = P(b = +1|x, u)$ associated with each phrase-pair (x, u) in the phrase-pair inventory. This is the probability that the phrase-pair (x, u) appears out of order in the transformed phrase sequence.

We now describe the MJ-1 WFST. In the presentation, we use upper-case letters to denote the English phrases (u_k) and lower-case letters to denote the French phrases (x_k and y_k).

The PPI for this example is given in Table 1.

English	French	Parameters	
u	x	$P(x\|u)$	$\beta_1(x, u)$
A	a	0.5	0.2
A	d	0.5	0.2
B	b	1.0	0.4
C	c	1.0	0.3
D	d	1.0	0.8

Table 1: Example phrase-pair inventory with translation and reordering probabilities.

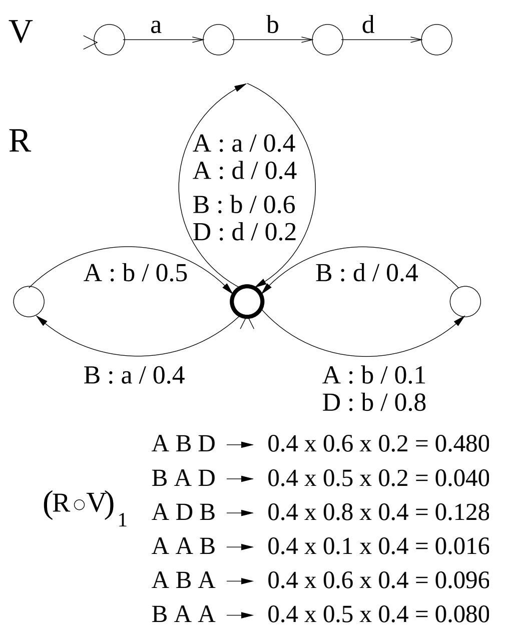
The input to the WFST (Fig 4) is a lattice of French phrase sequences derived from the French sentence to be translated. The outputs are the corresponding English phrase sequences. Note that the reordering is performed on the English side.

The WFST is constructed by adding a self-loop for each French phrase in the input lattice, and a 2-arc path for every pair of adjacent French phrases in the lattice. The WFST incorporates the translation model $P(x|u)$ and the reordering model $P(b|x, u)$. The score on a self-loop with labels (u, x) is $P(x|u) \times (1 - \beta_1(x, u))$; on a 2-arc path with labels (u_1, x_1) and (u_2, x_2), the score on the 1st arc is $P(x_2|u_1) \times \beta_1(x_2, u_1)$ and on the 2nd arc is $P(x_1|u_2)$.

In this example, the input to this transducer is a single French phrase sequence $V : a, b, c$. We perform the WFST composition $R \circ V$, project the result on the input labels, and remove the epsilons to form the acceptor $(R \circ V)_1$ which contains the six English phrase sequences (Fig 4).

Translation Given a French sentence, a lattice of translations is obtained using the weighted finite state composition: $\mathcal{T} = G \circ W \circ R \circ \Phi \circ \Omega \circ T$. The most-likely translation is obtained as the path with the highest probability in $\mathcal{T}$.

Alignment Given a sentence-pair (E, F), a lattice of phrase alignments is obtained by the finite state composition: $\mathcal{B} = S \circ W \circ R \circ \Phi \circ \Omega \circ T$, where



A B D → 0.4 x 0.6 x 0.2 = 0.480
B A D → 0.4 x 0.5 x 0.2 = 0.040
A D B → 0.4 x 0.8 x 0.4 = 0.128
A A B → 0.4 x 0.1 x 0.4 = 0.016
A B A → 0.4 x 0.6 x 0.4 = 0.096
B A A → 0.4 x 0.5 x 0.4 = 0.080

Figure 4: WFST for the MJ-1 model.

S is an acceptor for the English sentence E, and T is an acceptor for the French sentence F. The Viterbi alignment is found as the path with the highest probability in $\mathcal{B}$. The WFST composition gives the word-to-word alignments between the sentences. However, to obtain the phrase alignments, we need to construct additional FSTs not described here.

2.3 Reordering WFST for MJ-2

MJ-2 reordering restricts the maximum allowable jump to 2 phrases and also insists that the reordering take place within a window of 3 phrases. This latter condition implies that for an input sequence $\{a, b, c, d\}$, we disallow the three output sequences: $\{b, d, a, c;\ c, a, d, b;\ c, d, a, b;\}$. In the MJ-2 finite state machine, a given history b_1^{k-1} can lead to one of the six states in Fig 5.

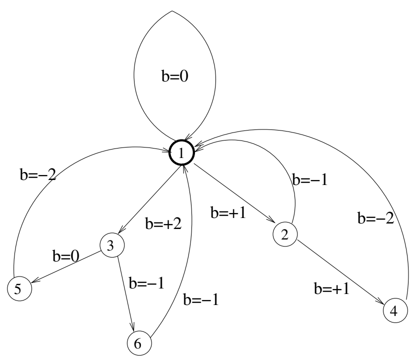


Figure 5: Phrase reordering process for MJ-2.

The jump probability of Eqn 5 becomes

$$P(b_k|x_k, u_k, \phi_{k-1}) =$$

$$\begin{cases} \beta_1(x_k, u_k) & b_k = 1, \phi_{k-1} = 1 \\ \beta_2(x_k, u_k) & b_k = 2, \phi_{k-1} = 1 \\ \begin{cases} 1 - \beta_1(x_k, u_k) \\ \quad -\beta_2(x_k, u_k) \end{cases} & b_k = 0, \phi_{k-1} = 1 \end{cases} \tag{7}$$

$$\begin{cases} \beta_1(x_k, u_k) & b_k = 1, \phi_{k-1} = 2 \\ 1 - \beta_1(x_k, u_k) & b_k = -1, \phi_{k-1} = 2 \end{cases} \tag{8}$$

$$\begin{cases} 0.5 & b_k = 0, \phi_{k-1} = 3 \\ 0.5 & b_k = -1, \phi_{k-1} = 3. \end{cases} \tag{9}$$

$$\begin{cases} 1 & b_k = -2, \phi_{k-1} = 4 \end{cases} \tag{10}$$

$$\begin{cases} 1 & b_k = -2, \phi_{k-1} = 5 \end{cases} \tag{11}$$

$$\begin{cases} 1 & b_k = -1, \phi_{k-1} = 6 \end{cases} \tag{12}$$

We note that the distributions (Eqns 7 and 8) are based on two parameters $\beta_1(x, u)$ and $\beta_2(x, u)$ for each phrase-pair (x, u).

Suppose the input is a phrase sequence a, b, c, the MJ-2 model (Fig 5) allows 6 possible reorderings: $a, b, c; a, c, b; b, a, c; b, c, a; c, a, b; c, b, a$. The distribution Eqn 9 ensures that the sequences b, c, a and c, b, a are assigned equal probability. The distributions in Eqns 10-12 ensure that the maximum jump is 2 phrases and the reordering happens within a window of 3 phrases. By insisting that the process start and end at state 1 (Fig 5), we ensure that the model is not deficient. A WFST implementing the MJ-2 model can be easily constructed for both phrase alignment and translation, following the construction described for the MJ-1 model.

3 Estimation of the Reordering Models

The Translation Template Model relies on an inventory of target language phrases and their source language translations. Our goal is to estimate the reordering model parameters $P(b|x, u)$ for each phrase-pair (x, u) in this inventory. However, when translating a given test set, only a subset of the phrase-pairs is needed. Although there may be an advantage in estimating the model parameters under an inventory that covers all the training bitext, we fix the phrase-pair inventory to cover only the phrases on the test set. Estimation of the reordering model parameters over the training bitext is then performed under this test-set specific inventory.

We employ the EM algorithm to obtain Maximum Likelihood (ML) estimates of the reordering model parameters. Applying EM to the MJ-1 reordering model gives the following ML parameter estimates for each phrase-pair (u, x).

$$\hat{\beta}_1(x,u) = \frac{C_{x,u}(0,+1)}{C_{x,u}(0,+1) + C_{x,u}(0,0)}. \quad (13)$$

$C_{x,u}(\phi, b)$ is defined for $\phi = 1, 2$ and $b = -1, 0, +1$. Any permissible phrase alignment of a sentence pair corresponds to a b_1^K sequence, which in turn specifies a ϕ_1^K sequence. $C_{x,u}(\phi, b)$ is the expected number of times the phrase-pair x, u is aligned with a jump of b phrases when the jump history is ϕ. We do not use full EM but a Viterbi training procedure that obtains the counts for the best (Viterbi) alignments. If a phrase-pair (x, u) is never seen in the Viterbi alignments, we back-off to a flat parameter $\beta_1(x, u) = 0.05$.

The ML parameter estimates for the MJ-2 model are given in Table 2, with $C_{x,u}(\phi, b)$ defined similarly. In our training scenario, we use WFST operations to obtain Viterbi phrase alignments of the training bitext where the initial reordering model parameters ($\beta_0(x, u)$) are set to a uniform value of 0.05. The counts $C_{x,u}(s, b)$ are then obtained over the phrase alignments. Finally the ML estimates of the parameters are computed using Eqn 13 (MJ-1) or Eqn 14 (MJ-2). We will refer to the Viterbi trained models as MJ-1 VT and MJ-2 VT. Table 3 shows the MJ-1 VT parameters for some example phrase-pairs in the Arabic-English (A-E) task.

u	x	$\beta_1(x,u)$
which_is_the_closest	Aqrb	1.0
international_trade	tjArp_EAlmyp	0.8
the_foreign_ministry	wzArp_xArjyp	0.6
arab_league	jAmEp_dwl_Erbyp	0.4

Table 3: MJ-1 parameters for A-E phrase-pairs.

To validate alignment under a PPI, we measure performance of the TTM word alignments on French-English (500 sent-pairs) and Chinese-English (124 sent-pairs) (Table 4). As desired, the Alignment Recall (AR) and Alignment Error Rate (AER) improve modestly while Alignment Precision (AP) remains constant. This suggests that the models allow more words to be aligned and thus improve the recall; MJ-2 gives a further improvement in AR and AER relative to MJ-1. Alignment precision depends on the quality of the word alignments within the phrase-pairs and does not change much by allowing phrase reordering. This experiment validates the estimation procedure based on the phrase alignments; however, we do not advocate the use of TTM as an alternate word alignment technique.

Reordering	Metrics (%)					
	Frn-Eng			Chn-Eng		
	AP	AR	AER	AP	AR	AER
None	94.2	84.8	10.0	85.1	47.1	39.3
MJ-1 VT	94.1	86.8	9.1	85.3	49.4	37.5
MJ-2 VT	93.9	87.4	8.9	85.3	50.9	36.3

Table 4: Alignment Performance with Reordering.

4 Translation Experiments

We perform our translation experiments on the large data track of the NIST Arabic-to-English (A-E) and Chinese-to-English (C-E) MT tasks; we report results on the NIST 2002, 2003, and 2004 evaluation test sets [1].

4.1 Exploratory Experiments

In these experiments the training data is restricted to FBIS bitext in C-E and the news bitexts in A-E. The bitext consists of chunk pairs aligned at sentence and sub-sentence level (Deng et al., 2004). In A-E, the training bitext consists of 3.8M English words, 3.2M Arabic words and 137K chunk pairs. In C-E, the training bitext consists of 11.7M English words, 8.9M Chinese words and 674K chunk pairs.

Our Chinese text processing consists of word segmentation (using the LDC segmenter) followed by grouping of numbers. For Arabic our text processing consisted of a modified Buckwalter analysis (LDC2002L49) followed by post processing to separate conjunctions, prepostions and pronouns, and Al-/w- deletion. The English text is processed using a simple tokenizer based on the text processing utility available in the the NIST MT-eval toolkit.

The Language Model (LM) training data consists of approximately 400M words of English text derived from Xinhua and AFP (English Gigaword), the English side of FBIS, the UN and A-E News texts, and the online archives of The People's Daily.

Table 5 gives the performance of the MJ-1 and MJ-2 reordering models when translation is performed using a 4-gram LM. We report performance on the 02, 03, 04 test sets and the combined test set

[1] `http://www.nist.gov/speech/tests/mt/`

$$\hat{\beta}_1(x,u) = \frac{C_{x,u}(1,+1) + C_{x,u}(2,+1)}{C_{x,u}(1,+1) + C_{x,u}(1,0) + C_{x,u}(1,+2) + C_{x,u}(2,+1) + C_{x,u}(2,-1)}$$

$$\hat{\beta}_2(x,u) = \frac{(C_{x,u}(1,0) + C_{x,u}(2,-1) + C_{x,u}(1,+2))C_{x,u}(1,+2)}{(C_{x,u}(1,+1) + C_{x,u}(1,0) + C_{x,u}(1,+2) + C_{x,u}(2,+1) + C_{x,u}(2,-1))(C_{x,u}(1,+2) + C_{x,u}(1,0))}$$

Table 2: ML parameter estimates for MJ-2 model.

Reordering	BLEU (%)							
	Arabic-English				Chinese-English			
	02	03	04	ALL	02	03	04	ALL
None	37.5	40.3	36.8	37.8 ± 0.6	24.2	23.7	26.0	25.0 ± 0.5
MJ-1 flat	40.4	43.9	39.4	40.7 ± 0.6	25.7	24.5	27.4	26.2 ± 0.5
MJ-1 VT	41.3	44.8	40.3	41.6 ± 0.6	25.8	24.5	27.8	26.5 ± 0.5
MJ-2 flat	41.0	44.4	39.7	41.1 ± 0.6	26.4	24.9	27.7	26.7 ± 0.5
MJ-2 VT	41.7	45.3	40.6	42.0 ± 0.6	26.5	24.9	27.9	26.8 ± 0.5

Table 5: Performance of MJ-1 and MJ-2 reordering models with a 4-gram LM.

(ALL=02+03+04). For the combined set (ALL), we also show the 95% BLEU confidence interval computed using bootstrap resampling (Och, 2003).

Row 1 gives the performance when no reordering model is used. The next two rows show the influence of the MJ-1 reordering model; in row 2, a flat probability of $\beta_1(x,u) = 0.05$ is used for all phrase-pairs; in row 3, a reordering probability is estimated for each phrase-pair using Viterbi Training (Eqn 13). The last two rows show the effect of the MJ-2 reordering model; row 4 uses flat probabilities ($\beta_1(x,u) = 0.05, \beta_2(x,u) = 0.01$) for all phrase-pairs; row 5 applies reordering probabilities estimating with Viterbi Training for each phrase-pair (Table 2).

On both language-pairs, we observe that reordering yields significant improvements. The gains from phrase reordering are much higher on A-E relative to C-E; this could be related to the fact that the word order differences between English and Arabic are much higher than the differences between English and Chinese. MJ-1 VT outperforms flat MJ-1 showing that there is value in estimating the reordering parameters from bitext. Finally, the MJ-2 VT model performs better than the flat MJ-2 model, but only marginally better than the MJ-1 VT model. Therefore estimation does improve the MJ-2 model but allowing reordering beyond a window of 1 phrase is not useful when translating either Arabic or Chinese into English in this framework.

The flat MJ-1 model outperforms the no-reordering case and the flat MJ-2 model is better than the flat MJ-1 model; we hypothesize that phrase reordering increases search space of translations that allows the language model to select a higher quality hypothesis. This suggests that these models of phrase reordering actually require strong language models to be effective. We now investigate the interaction between language models and reordering.

Our goal here is to measure translation performance of reordering models over variable span n-gram LMs (Table 6). We observe that both MJ-1 and MJ-2 models yield higher improvements under higher order LMs: e.g. on A-E, gains under 3g (3.6 BLEU points on MJ-1, 0.2 points on MJ-2) are higher than the gains with 2g (2.4 BLEU points on MJ-1, 0.1 points on MJ-2).

Reordering	BLEU (%)					
	A-E			C-E		
	2g	3g	4g	2g	3g	4g
None	21.0	36.8	37.8	16.1	24.8	25.0
MJ-1 VT	23.4	40.4	41.6	16.2	25.9	26.5
MJ-2 VT	23.5	40.6	42.0	16.0	26.1	26.8

Table 6: Reordering with variable span n-gram LMs on Eval02+03+04 set.

We now measure performance of the reordering models across the three test set genres used in the NIST 2004 evaluation: news, editorials, and speeches. On A-E, MJ-1 and MJ-2 yield larger improvements on News relative to the other genres; on C-E, the gains are larger on Speeches and Editorials relative to News. We hypothesize that the Phrase-Pair Inventory, reordering models and language models could all have been biased away from the test set due to the training data. There may also be less movement across these other genres.

Reordering	BLEU (%)					
	A-E			C-E		
	News	Eds	Sphs	News	Eds	Sphs
None	41.1	30.8	33.3	23.6	25.9	30.8
MJ-1 VT	45.6	32.6	35.7	24.8	27.8	33.3
MJ-2 VT	46.2	32.7	35.5	24.8	27.8	33.7

Table 7: Performance across Eval 04 test genres.

	BLEU (%)					
	Arabic-English			Chinese-English		
Reordering	02	03	04n	02	03	04n
None	40.2	42.3	43.3	28.9	27.4	27.3
MJ-1 VT	43.1	45.0	45.6	30.2	28.2	28.9
MET-Basic	44.8	47.2	48.2	31.3	30.3	30.3
MET-IBM1	45.2	48.2	49.7	31.8	30.7	31.0

Table 8: Translation Performance on Large Bitexts.

4.2 Scaling to Large Bitext Training Sets

We here describe the integration of the phrase reordering model in an MT system trained on large bitexts. The text processing and language models have been described in § 4.1. Alignment Models are trained on all available bitext (7.6M chunk pairs/207.4M English words/175.7M Chinese words on C-E and 5.1M chunk pairs/132.6M English words/123.0M Arabic words on A-E), and word alignments are obtained over the bitext. Phrase-pairs are then extracted from the word alignments (Koehn et al., 2003). MJ-1 model parameters are estimated over all bitext on A-E and over the non-UN bitext on C-E. Finally we use Minimum Error Training (MET) (Och, 2003) to train log-linear scaling factors that are applied to the WFSTs in Equation 1. 04news (04n) is used as the MET training set.

Table 8 reports the performance of the system. Row 1 gives the performance without phrase reordering and Row 2 shows the effect of the MJ-1 VT model. The MJ-1 VT model is used in an initial decoding pass with the four-gram LM to generate translation lattices. These lattices are then rescored under parameters obtained using MET (MET-basic), and 1000-best lists are generated. The 1000-best lists are augmented with IBM Model-1 (Brown et al., 1993) scores and then rescored with a second set of MET parameters. Rows 3 and 4 show the performance of the MET-basic and MET-IBM1 models.

We observe that the maximum likelihood phrase reordering model (MJ-1 VT) yields significantly improved translation performance relative to the monotone phrase order translation baseline. This confirms the translation performance improvements found over smaller training bitexts.

We also find additional gains by applying MET to optimize the scaling parameters that are applied to the WFST component distributions within the TTM (Equation 1). In this procedure, the scale factor applied to the MJ-1 VT Phrase Translation and Reordering component is estimated along with scale factors applied to the other model components; in other words, the ML-estimated phrase reordering model itself is not affected by MET, but the likelihood that it assigns to a phrase sequence is scaled by a single, discriminatively optimized weight. The improvements from MET (see rows MET-Basic and MET- IBM1) demonstrate that the MJ-1 VT reordering models can be incorporated within a discriminative optimized translation system incorporating a variety of models and estimation procedures.

5 Discussion

In this paper we have described local phrase reordering models developed for use in statistical machine translation. The models are carefully formulated so that they can be implemented as WFSTs, and we show how the models can be incorporated into the Translation Template Model to perform phrase alignment and translation using standard WFST operations. Previous approaches to WFST-based reordering (Knight and Al-Onaizan, 1998; Kumar and Byrne, 2003; Tsukada and Nagata, 2004) constructed permutation acceptors whose state spaces grow exponentially with the length of the sentence to be translated. As a result, these acceptors have to be pruned heavily for use in translation. In contrast, our models of local phrase movement do not grow explosively and do not require any pruning or approximation in their construction. In other related work, Bangalore and Ricardi (2001) have trained WFSTs for modeling reordering within translation; their WFST parses word sequences into trees containing reordering information, which are then checked for well-formed brackets. Unlike this approach, our model formulation does not use a tree representation and also ensures that the output sequences are valid permutations of input phrase sequences; we emphasize again that the probability distribution induced over reordered phrase sequences is not degenerate.

Our reordering models do resemble those of (Tillmann, 2004; Tillmann and Zhang, 2005) in that we treat the reordering as a sequence of jumps relative to the original phrase sequence, and that the likelihood of the reordering is assigned through phrase-pair specific parameterized models. We note that our implementation allows phrase reordering beyond simply a 1-phrase window, as was done by Tillmann. More importantly, our model implements a generative model of phrase reordering which can be incorporated directly into a generative model of the overall translation process. This allows us to perform 'embedded' EM-style parameter estimation, in which the parameters of the phrase reordering model are estimated using statistics gathered under the complete model that will actually be used in translation. We believe that this estimation of model parameters directly from phrase alignments obtained under the phrase translation model is a novel contribution; prior approaches derived the parameters of the reordering models from word aligned bitext, e.g. within the phrase pair extraction procedure.

We have shown that these models yield improvements in alignment and translation performance on Arabic-English and Chinese-English tasks, and that the reordering model can be integrated into large evaluation systems. Our experiments show that discriminative training procedures such Minimum Error Training also yield additive improvements by tuning TTM systems which incorporate ML-trained reordering models. This is essential for integrating our reordering model inside an evaluation system, where a variety of techniques are applied simultaneously.

The MJ-1 and MJ-2 models are extremely simple models of phrase reordering. Despite their simplicity, these models provide large improvements in BLEU score when incorporated into a monotone phrase order translation system. Moreover, they can be used to produced translation lattices for use by more sophisticated reordering models that allow longer phrase order movement. Future work will build on these simple structures to produce more powerful models of word and phrase movement in translation.

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.

Y. Deng, S. Kumar, and W. Byrne. 2004. Bitext chunk alignment for statistical machine translation. In *Research Note, Center for Language and Speech Processing, Johns Hopkins University*.

K. Knight and Y. Al-Onaizan. 1998. Translation with finite-state devices. In *AMTA*, pages 421–437, Langhorne, PA, USA.

K. Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics, Squibs & Discussion*, 25(4).

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pages 127–133, Edmonton, Canada.

S. Kumar and W. Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *HLT-NAACL*, pages 142–149, Edmonton, Canada.

S. Kumar, Y. Deng, and W. Byrne. 2005. A weighted finite state transducer translation template model for statistical machine translation. *Journal of Natural Language Engineering*, 11(4).

F. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *EMNLP-VLC*, pages 20–28, College Park, MD, USA.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, Sapporo, Japan.

C. Tillmann and T. Zhang. 2005. A localized prediction model for statistical machine translation. In *ACL*, Ann Arbor, Michigan, USA.

C. Tillmann. 2004. A block orientation model for statistical machine translation. In *HLT-NAACL*, Boston, MA, USA.

H. Tsukada and M. Nagata. 2004. Efficient decoding for statistical machine translation with a fully expanded WFST model. In *EMNLP*, Barcelona, Spain.

S. Vogel. 2003. SMT Decoder Dissected: Word Reordering. In *NLPKE*, Beijing, China.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *ACL*, pages 152–158, Santa Cruz, CA, USA.

R. Zens and H. Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *ACL*, pages 144–151, Sapporo, Japan.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *COLING*, pages 205–211, Boston, MA, USA.

HMM Word and Phrase Alignment for Statistical Machine Translation

Yonggang Deng[1] , **William Byrne**[1,2]
Center for Language and Speech Processing, Johns Hopkins University
Baltimore, MD 21210, USA [1]
Machine Intelligence Lab, Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, UK [2]
dengyg@jhu.edu , wjb31@cam.ac.uk

Abstract

HMM-based models are developed for the alignment of words and phrases in bitext. The models are formulated so that alignment and parameter estimation can be performed efficiently. We find that Chinese-English word alignment performance is comparable to that of IBM Model-4 even over large training bitexts. Phrase pairs extracted from word alignments generated under the model can also be used for phrase-based translation, and in Chinese to English and Arabic to English translation, performance is comparable to systems based on Model-4 alignments. Direct phrase pair induction under the model is described and shown to improve translation performance.

1 Introduction

Describing word alignment is one of the fundamental goals of Statistical Machine Translation (SMT). Alignment specifies how word order changes when a sentence is translated into another language, and given a sentence and its translation, alignment specifies translation at the word level. It is straightforward to extend word alignment to phrase alignment: two phrases align if their words align.

Deriving phrase pairs from word alignments is now widely used in phrase-based SMT. Parameters of a statistical word alignment model are estimated from bitext, and the model is used to generate word alignments over the same bitext. Phrase pairs are extracted from the aligned bitext and used in the SMT system. With this approach the quality of the underlying word alignments can have a strong influence on phrase-based SMT system performance. The common practice therefore is to extract phrase pairs from the best attainable word alignments. Currently, Model-4 alignments (Brown and others, 1993) as produced by GIZA++ (Och and Ney, 2000) are often the best that can be obtained, especially with large bitexts.

Despite its modeling power and widespread use, Model-4 has shortcomings. Its formulation is such that maximum likelihood parameter estimation and bitext alignment are implemented by approximate, hill-climbing, methods. Consequently parameter estimation can be slow, memory intensive, and difficult to parallelize. It is also difficult to compute statistics under Model-4. This limits its usefulness for modeling tasks other than the generation of word alignments.

We describe an HMM alignment model developed as an alternative to Model-4. In the word alignment and phrase-based translation experiments to be presented, its performance is comparable or improved relative to Model-4. Practically, we can train the model by the Forward-Backward algorithm, and by parallelizing estimation, we can control memory usage, reduce the time needed for training, and increase the bitext used for training. We can also compute statistics under the model in ways not practical with Model-4, and we show the value of this in the extraction of phrase pairs from bitext.

2 HMM Word and Phrase Alignment

Our goal is to develop a generative probabilistic model of Word-to-Phrase (WtoP) alignment. We start with an l-word source sentence $\mathbf{e} = e_1^l$, and an

m-word target sentence $\mathbf{f} = f_1^m$, which is realized as a sequence of K phrases: $\mathbf{f} = v_1^K$.

Each phrase is generated as a translation of one source word, which is determined by the alignment sequence a_1^K: $e_{a_k} \rightarrow v_k$. The length of each phrase is specified by the process ϕ_1^K, which is constrained so that $\sum_{k=1}^K \phi_k = m$.

We also allow target phrases to be inserted, i.e. to be generated by a NULL source word. For this, we define a binary hallucination sequence h_1^K : if $h_k = 0$, then NULL $\rightarrow v_k$; if $h_k = 1$ then $e_{a_k} \rightarrow v_k$.

With all these quantities gathered into an alignment $\mathbf{a} = (\phi_1^K, a_1^K, h_1^K, K)$, the modeling objective is to realize the conditional distribution $P(\mathbf{f}, \mathbf{a}|\mathbf{e})$. With the assumption that $P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0$ if $\mathbf{f} \neq v_1^K$, we write $P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = P(v_1^K, K, a_1^K, h_1^K, \phi_1^K|\mathbf{e})$ and

$$\begin{aligned} &P(v_1^K, K, a_1^K, h_1^K, \phi_1^K|\mathbf{e}) \\ &\quad = \epsilon(m|l) \times P(K|m, \mathbf{e}) \\ &\qquad \times P(a_1^K, \phi_1^K, h_1^K|K, m, \mathbf{e}) \\ &\qquad\quad \times P(v_1^K|a_1^K, h_1^K, \phi_1^K, K, m, \mathbf{e}) \end{aligned}$$

We now describe the component distributions.

Sentence Length $\epsilon(m|l)$ determines the target sentence length. It is not needed during alignment, where sentence lengths are known, and is ignored.

Phrase Count $P(K|m, \mathbf{e})$ specifies the number of target phrases. We use a simple, single parameter distribution, with $\eta = 8.0$ throughout

$$P(K|m, \mathbf{e}) = P(K|m, l) \propto \eta^K$$

Word-to-Phrase Alignment Alignment is a Markov process that specifies the lengths of phrases and their alignment with source words

$$\begin{aligned} &P(a_1^K, h_1^K, \phi_1^K|K, m, \mathbf{e}) \\ &= \prod_{k=1}^K P(a_k, h_k, \phi_k|a_{k-1}, \phi_{k-1}, \mathbf{e}) \\ &= \prod_{k=1}^K p(a_k|a_{k-1}, h_k; l)\, d(h_k)\, n(\phi_k; e_{a_k}) \end{aligned}$$

The actual word-to-phrase alignment (a_k) is a first-order Markov process, as in HMM-based word-to-word alignment (Vogel et al., 1996). It necessarily depends on the hallucination variable

$$\begin{aligned} &p(a_j|a_{j-1}, h_j; l) \\ &\quad = \begin{cases} 1 & a_j = a_{j-1},\ h_j = 0 \\ 0 & a_j \neq a_{j-1},\ h_j = 0 \\ a(a_j|a_{j-1}; l) & h_j = 1 \end{cases} \end{aligned}$$

This formulation allows target phrases to be inserted without disrupting the Markov dependencies of phrases aligned to actual source words.

The phrase length model $n(\phi; e)$ gives the probability that a word e produces a phrase with ϕ words in the target language; $n(\phi; e)$ is defined for $\phi = 1, \cdots, N$. The hallucination process is a simple i.i.d. process, where $d(0) = p_0$, and $d(1) = 1 - p_0$.

Word-to-Phrase Translation The translation of words to phrases is given as

$$P(v_1^K|a_1^K, h_1^K, \phi_1^K, K, m, \mathbf{e}) = \prod_{k=1}^K p(v_k|e_{a_k}, h_k, \phi_k)$$

We introduce the notation $v_k = v_k[1], \ldots, v_k[\phi_k]$ and a dummy variable x_k (for phrase insertion) :

$$x_k = \begin{cases} e_{a_k} & h_k = 1 \\ \text{NULL} & h_k = 0 \end{cases}$$

We define two models of word-to-phrase translation. This simplest is based on context-independent word-to-word translation

$$p(v_k|e_{a_k}, h_k, \phi_k) = \prod_{j=1}^{\phi_k} t(v_k[j] \,|\, x_k)$$

We also define a model that captures foreign word context with *bigram translation probabilities*

$$\begin{aligned} &p(v_k|e_{a_k}, h_k, \phi_k) \\ &\quad = t(v_k[1] \,|\, x_k) \prod_{j=2}^{\phi_k} t_2(v_k[j] \,|\, v_k[j-1], x_k) \end{aligned}$$

Here, $t(f|e)$ is the usual context independent word-to-word translation probability. The bigram translation probability $t_2(f|f', e)$ specifies the likelihood that target word f is to follow f' in a phrase generated by source word e.

2.1 Properties of the Model and Prior Work

The formulation of the WtoP alignment model was motivated by both the HMM word alignment model (Vogel et al., 1996) and IBM Model-4 with the goal of building on the strengths of each.

The relationship with the word-to-word HMM alignment model is straightforward. For example, constraining the phrase length component $n(\phi; e)$ to permit only phrases of one word would give a word-to-word HMM alignment model. The extensions introduced are the phrase count, and the phrase length models, and the bigram translation distribution. The hallucination process is motivated by the use of NULL alignments into Markov alignment models as done by (Och and Ney, 2003).

The phrase length model is motivated by Toutanova et al. (2002) who introduced 'stay' probabilities in HMM alignment as an alternative to word fertility. By comparison, Word-to-Phrase HMM alignment models contain detailed models of state occupancy, motivated by the IBM fertility model, which are more powerful than a single staying parameter. In fact, the WtoP model is a segmental Hidden Markov Model (Ostendorf et al., 1996), in which states emit observation sequences.

Comparison with Model-4 is less straightforward. The main features of Model-4 are NULL source words, source word fertility, and the distortion model. The WtoP alignment model includes the first two of these. However distortion, which allows hypothesized words to be distributed throughout the target sentence, is difficult to incorporate into a model that supports efficient DP-based search. We preserve efficiency in the WtoP model by insisting that target words form connected phrases; this is not as general as Model-4 distortion. This weakness is somewhat offset by a more powerful (Markov) alignment process as well as by the phrase count distribution. Despite these differences, the WtoP alignment model and Model-4 allow similar alignments. For example, in Fig. 1, Model-4 would allow f_1, f_3, and f_4 to be generated by e_1 with a fertility of 3. Under the WtoP model, e_1 could generate f_1 and $f_3 f_4$ with phrase lengths 1 and 2, respectively: source words can generate more than one phrase.

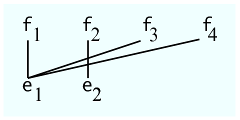


Figure 1: Word-to-Word and Word-to-Phrase Links

This alignment could also be generated via four single word foreign phrases. The balance between word-to-word and word-to-phrase alignments is set by the phrase count distribution parameter η. As η increases, alignments with shorter phrases are favored, and for very large η the model allows only word-to-word alignments (see Fig. 2). Although the WtoP alignment model is more complex than the word-to-word HMM alignment model, the Baum-Welch and Viterbi algorithms can still be used. Word-to-word alignments are generated by the Viterbi algorithm: $\hat{\mathbf{a}} = \operatorname{argmax}_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})$; if $e_{a_k} \rightarrow v_k$, e_{a_k} is linked to all the words in v_k.

The bigram translation probability relies on word context, known to be helpful in translation (Berger et al., 1996), to improve the identification of target phrases. As an example, f is the Chinese word for "world trade center". Table 1 shows how the likelihood of the correct English phrase is improved with bigram translation probabilities; this example is from the C→E, N=4 system of Table 2.

Model	unigram	bigram
$P(\text{world}\|f)$	0.06	0.06
$P(\text{trade}\|\text{world}, f)$	0.06	0.99
$P(\text{center}\|\text{trade}, f)$	0.06	0.99
$P(\text{world trade center}\|f, 3)$	0.0002	0.0588

Table 1: Context in Bigram Phrase Translation.

There are of course much prior work in translation that incorporates phrases. Sumita et al. (2004) develop a model of phrase-to-phrase alignment, which while based on HMM alignment process, appears to be deficient. Marcu and Wong (2002) propose a model to learn lexical correspondences at the phrase level. To our knowledge, ours is the first non-syntactic model of bitext alignment (as opposed to translation) that links words and phrases.

3 Embedded Alignment Model Estimation

We now discuss estimation of the WtoP model parameters by the EM algorithm. Since the WtoP model can be treated as an HMM with a very complex state space, it is straightforward to apply Baum-

Welch parameter estimation. We show the forward recursion as an example.

Given a sentence pair (e_1^l, f_1^m), the forward probability $\alpha_j(i, \phi)$ is defined as the probability of generating the first j target words with the added condition that the target words $f_{j-\phi+1}^j$ form a phrase aligned to source word e_i. It can be calculated recursively (omitting the hallucination process, for simplicity) as

$$\alpha_j(i,\phi) = \Big\{ \sum_{i',\phi'} \alpha_{j-\phi}(i',\phi') a(i|i',l) \Big\} \cdot \eta$$

$$\cdot\, n(\phi; e_i) \cdot t(f_{j-\phi+1}|e_i) \;\cdot \prod_{j'=j-\phi+2}^{j} t_2(f_{j'}|e_i)$$

This recursion is over a trellis of $l(N+1)m$ nodes.

Models are trained from a flat-start. We begin with 10 iterations of EM to train Model-1, followed by 5 EM iterations to train Model-2 (Brown and others, 1993). We initialize the parameters of the word-to-word HMM alignment model by collecting word alignment counts from the Model-2 Viterbi alignments, and refine the word-to-word HMM alignment model by 5 iterations of the Baum-Welch algorithm. We increase the order of the WtoP model (N) from 2 to the final value in increments of 1, by performing 5 Baum Welch iterations at each step. At the final value of N, we introduce the bigram translation probability; we use Witten-Bell smoothing (1991) as a backoff strategy for t_2, and other strategies are possible.

4 Bitext Word Alignment

We now investigate bitext word alignment performance. We start with the FBIS Chinese/English parallel corpus which consists of approx. 10M English/7.5M Chinese words. The Chinese side of the corpus is segmented into words by the LDC segmenter[1]. The alignment test set consists of 124 sentences from the NIST 2001 dry-run MT-eval[2] set that are manually word aligned.

We first analyze the distribution of word links within these manual alignments. Of the Chinese words which are aligned to more than one English words, 82% of these words align with consecutive English words (phrases). In the other direction, among all English words which are aligned to multiple Chinese words, 88% of these align to Chinese phrases. In this collection, at least, word-to-phrase alignments are plentiful.

Alignment performance is measured by the Alignment Error Rate (AER) (Och and Ney, 2003)

$$AER(B; B') = 1 - 2 \times |B \cap B'|/(|B'| + |B|)$$

where B is a set reference word links, and B' are the word links generated automatically.

AER gives a general measure of word alignment quality. We are also interested in how the model performs over the word-to-word and word-to-phrase alignments it supports. We split the reference alignments into two subsets: B_{1-1} contains word-to-word reference links (e.g. 1⟶1 in Fig 1); and B_{1-N} contains word-to-phrase reference links (e.g. 1⟶3, 1⟶4 in Fig 1); The automatic alignment B' is partitioned similarly. We define additional AERs: $AER_{1-1} = AER(B_{1-1}, B'_{1-1})$, and $AER_{1-N} = AER(B_{1-N}, B'_{1-N})$, which measure word-to-word and word-to-phrase alignment, separately.

Table 2 presents the three AER measurements for

Model	AER_{1-1}	AER_{1-N}	AER
	C⟶E		
Model-4	37.9	68.3	37.3
HMM, N=1	42.8	72.9	42.0
HMM, N=2	38.3	71.2	38.1
HMM, N=3	37.4	69.5	37.8
HMM, N=4	37.1	69.1	37.8
+ bigram t-table	37.5	65.8	37.1
	E⟶C		
Model-4	42.3	87.2	45.0
HMM, N=1	45.0	90.6	47.2
HMM, N=2	42.7	87.5	44.5
+ bigram t-table	44.2	85.5	45.1

Table 2: FBIS Bitext Alignment Error Rate.

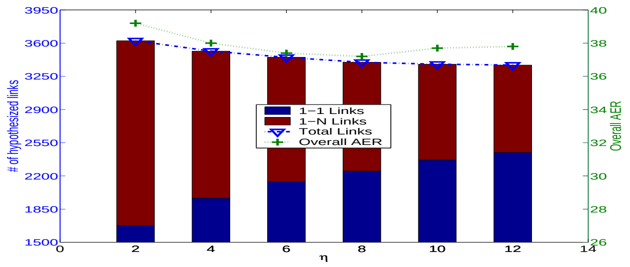


Figure 2: Balancing Word and Phrase Alignments

[1]http://www.ldc.upenn.edu/Projects/Chinese

[2]http://www.nist.gov/speech/tests/mt

the WtoP alignment models trained as described in Section 3. GIZA++ Model 4 alignment performance is also presented for comparison. We note first that the word-to-word HMM (N=1) alignment model is worse than Model 4, as expected. For the WtoP models in the C→E direction, we see reduced AER for phrases lengths up to 4, although in the E→C direction, AER is reduced only for phrases of length 2; performance for $N > 2$ is not reported.

In introducing the bigram phrase translation (the bigram t-table), there is a tradeoff between word-to-word and word-to-phrase alignment quality. As mentioned, the bigram t-table increases the likelihood of word-to-phrase alignments. In both translation directions, this reduces the AER_{1-N}. However, it also causes increases in AER_{1-1}, primarily due to a drop in recall: fewer word-to-word alignments are produced. For C→E, this is not severe enough to cause an overall AER increase; however, in E→C, AER does increase.

Fig. 2 (C→E, N=4) shows how the 1-1 and 1-N alignment behavior is balanced by the phrase count parameter. As η increases, the model favors alignments with more word-to-word links and fewer word-to-phrase links; the overall Alignment Error Rate (AER) suggests a good balance at $\eta = 8.0$.

After observing that the WtoP model performs as well as Model-4 over the FBIS C-E bitext, we investigated performance over these large bitexts :
- "NEWS" containing non-UN parallel Chinese/English corpora from LDC (mainly FBIS, Xinhua, Hong Kong, Sinorama, and Chinese Treebank).
- "NEWS+UN01-02" also including UN parallel corpora from the years 2001 and 2002.
- "ALL C-E" refers to all the C-E bitext available from LDC as of his submission; this consists of the NEWS corpora with the UN bitext from all years.

Over all these collections, WtoP alignment performance (Table 3) is comparable to that of Model-4. We do note a small degradation in the E→C WtoP alignments. It is quite possible that this one-to-many model suffers slightly with English as the source and Chinese as the target, since English sentences tend to be longer. Notably, simply increasing the amount of bitext used in training need not improve AER. However, larger aligned bitexts can give improved phrase pair coverage of the test set.

One of the desirable features of HMMs is that the

Bitext	English Words	Model	C→E	E→C
NEWS	71M	M-4	37.1	45.3
		WtoP	36.1	44.8
NEWS+ UN01-02	96M	M-4	36.1	43.4
		WtoP	36.4	44.2
ALL C-E	200M	WtoP	36.8	44.7

Table 3: AER Over Large C-E Bitexts.

Forward-Backward steps can be run in parallel: bitext is partitioned; the Forward-Backward algorithm is run over the subsets on different CPUs; statistics are merged to reestimate model parameters. Partitioning the bitext also reduces the memory usage, since different cooccurrence tables can be kept for each partition. With the "ALL C-E" bitext collection, a single set of WtoP models (C→E, N=4, bigram t-table) can be trained over 200M words of Chinese-English bitext by splitting training over 40 CPUs; each Forward-Backward process takes less than 2GB of memory and the training run finishes in five days. By contrast, the 96M English word NEWS+UN01-02 is about the largest C-E bitext over which we can train Model-4 with our GIZA++ configuration and computing infrastructure.

Based on these and other experiments, in this paper we set a maximum value of $N = 4$ for F→E; in E→F, we set N=2 and omit the bigram phrase translation probability; η is set to 8.0. We do not claim that this is optimal, however.

5 Phrase Pair Induction

A common approach to phrase-based translation is to extract an inventory of phrase pairs (PPI) from bitext (Koehn et al., 2003), For example, in the *phrase-extract* algorithm (Och, 2002), a word alignment $\hat{a}_1^m$ is generated over the bitext, and all word subsequences $e_{i_1}^{i_2}$ and $f_{j_1}^{j_2}$ are found that satisfy :

$$\hat{a}_1^m : \hat{a}_j \in [i_1, i_2] \text{ iff } j \in [j_1, j_2] \; . \qquad (1)$$

The PPI comprises all such phrase pairs $(e_{i_1}^{i_2}, f_{j_1}^{j_2})$.

The process can be stated slightly differently. First, we define a set of alignments :

$$A(i_1, i_2; j_1, j_2) = \{a_1^m : a_j \in [i_1, i_2] \text{ iff } j \in [j_1, j_2]\} \; .$$

If $\hat{a}_1^m \in A(i_1, i_2; j_1, j_2)$ then $(e_{i_1}^{i_2}, f_{j_1}^{j_2})$ form a phrase pair.

Viewed in this way, there are many possible alignments under which phrases might be paired, and the selection of phrase pairs need not be based on a single alignment. Rather than simply accepting a phrase pair $(e_{i_1}^{i_2}, f_{j_1}^{j_2})$ if the unique MAP alignment satisfies Equation 1, we can assign a probability to phrases occurring as translation pairs :

$$P(\mathbf{f},\, A(i_1,i_2;j_1,j_2\,)\,|\,\mathbf{e}) = \sum_{\mathbf{a}\,:\,a_1^m \in A(i_1,i_2;j_1,j_2\,)} P(\mathbf{f},\mathbf{a}|\mathbf{e})$$

For a fixed set of indices i_1, i_2, j_1, j_2, the quantity $P(\mathbf{f},\, A(i_1,i_2;j_1,j_2\,)\,|\,\mathbf{e})$ can be computed efficiently using a modified Forward algorithm. Since $P(\mathbf{f}|\mathbf{e})$ can also be computed by the Forward algorithm, the *phrase-to-phrase posterior distribution* $P(\,A(i_1,i_2;j_1,j_2\,)\,|\,\mathbf{f},\mathbf{e})$ is easily found.

PPI Induction Strategies In the *phrase-extract* algorithm (Och, 2002), the alignment $\hat{a}$ is generated as follows: Model-4 is trained in both directions (e.g. F→E and E→F); two sets of word alignments are generated by the Viterbi algorithm for each set of models; and the two alignments are merged. This forms a static aligned bitext. Next, all foreign word sequences up to a given length (here, 5 words) are extracted from the test set. For each of these, a phrase pair is added to the PPI if the foreign phrase can be found aligned to an English phrase under Eq 1. We refer to the result as the Model-4 Viterbi Phrase-Extract PPI.

Constructed in this way, the PPI is limited to phrase pairs which can be found in the Viterbi alignments. Some foreign phrases which do appear in the training bitext will not be included in the PPI because suitable English phrases cannot be found. To add these to the PPI we can use the phrase-to-phrase posterior distribution to find English phrases as candidate translations. This adds phrases to the Viterbi Phrase-Extract PPI and increase the test set coverage. A somewhat *ad hoc* PPI Augmentation algorithm is given to the right.

Condition (A) extracts phrase pairs based on the geometric mean of the E→F and F→E posteriors ($T_g = 0.01$ throughout). The threshold T_p selects additional phrase pairs under a more forgiving criterion: as T_p decreases, more phrase pairs are added and PPI coverage increases. Note that this algorithm is constructed specifically to improve a Viterbi PPI; it is certainly not the only way to extract phrase pairs under the phrase-to-phrase posterior distribution.

Once the PPI phrase pairs are set, the phrase translation probabilities are set based on the number of times each phrase pair is extracted from a sentence pair, i.e. from relative frequencies.

```
For each foreign phrase $v$ not in the Viterbi PPI :
  For all pairs $(f_1^m, e_1^l)$ and $j_1, j_2$ s.t. $f_{j_1}^{j_2} = v$ :
    For $1 \leq i_1 \leq i_2 \leq l$, find
      $f(i_1,i_2) = P_{F\to E}(\,A(i_1,i_2;j_1,j_2)\,|\,e_1^l, f_1^m)$
      $b(i_1,i_2) = P_{E\to F}(\,A(i_1,i_2;j_1,j_2)\,|\,e_1^l, f_1^m)$
      $g(i_1,i_2) = \sqrt{f(1_1,i_2)\,b(i_1,i_2)}$
      $(\hat{i}_1,\hat{i}_2) = \operatorname*{argmax}_{1\leq i_1,i_2\leq l} g(i_1,i_2)$ , and set $u = e_{\hat{i}_1}^{\hat{i}_2}$
    Add $(u,v)$ to the PPI if any of A, B, or C hold :
      $b(\hat{i}_1,\hat{i}_2) \geq T_g$ and $f(\hat{i}_1,\hat{i}_2) \geq T_g$    (A)
      $b(\hat{i}_1,\hat{i}_2) < T_g$ and $f(\hat{i}_1,\hat{i}_2) > T_p$    (B)
      $f(\hat{i}_1,\hat{i}_2) < T_g$ and $b(\hat{i}_1,\hat{i}_2) > T_p$    (C)
```

PPI Augmentation via Phrase-Posterior Induction

HMM-based models are often used if posterior distributions are needed. Model-1 can also be used in this way (Venugopal et al., 2003), although it is a relatively weak alignment model. By comparison, finding posterior distributions under Model-4 is difficult. The Word-to-Phrase alignment model appears not to suffer this tradeoff: it is a good model of word alignment under which statistics such as the phrase-to-phrase posterior can be calculated.

6 Translation Experiments

We evaluate the quality of phrase pairs extracted from the bitext through the translation performance of the Translation Template Model (TTM) (Kumar et al., 2005), which is a phrase-based translation system implemented using weighted finite state transducers. Performance is measured by BLEU (Papineni and others, 2001).

Chinese→English Translation We report performance on the NIST Chinese/English 2002, 2003 and 2004 (News only) MT evaluation sets. These consist of 878, 919, and 901 sentences, respectively. Each Chinese sentence has 4 reference translations.

We evaluate two C→E translation systems. The smaller system is built on the FBIS C-E bitext collection. The language model used for this system is a trigram word language model estimated with 21M

	V-PE Model	WtoP T_p	eval02		eval03		eval04		eval02		eval03		eval04	
			cvg	BLEU	cvg	BLEU	cvg	BLEU	cvg	BLEU	cvg	BLEU	cvg	BLEU
			FBIS C→E System						News A→E System					
1	M-4	-	20.1	23.8	17.7	22.8	20.2	23.0	19.5	36.9	21.5	39.1	18.5	40.0
2		0.7	24.6	24.6	21.4	23.7	24.6	23.7	23.8	37.6	26.6	40.2	22.4	40.3
3	WtoP	-	19.7	23.9	17.4	23.3	19.8	23.3	18.4	36.2	20.6	38.6	17.4	39.2
4		1.0	23.1	24.0	20.0	23.7	23.2	23.5	21.8	36.7	24.3	39.3	20.4	39.7
5		0.9	24.0	24.8	20.9	23.9	24.0	23.8	23.2	37.2	25.8	39.7	21.8	40.1
6		0.7	24.6	24.9	21.3	24.0	24.7	23.9	23.7	37.2	26.5	39.7	22.4	39.9
7		0.5	24.9	24.9	21.6	24.1	24.8	23.9	24.0	37.2	26.9	39.7	22.7	39.8
			Large C→E System						Large A→E System					
8	M-4	-	32.5	27.7	29.3	27.1	32.5	26.6	26.4	38.1	28.1	40.1	28.2	39.9
9	WtoP	-	30.6	27.9	27.5	27.0	30.6	26.4	24.8	38.1	26.6	40.1	26.7	40.6
10		0.7	38.2	28.2	32.3	27.3	37.1	26.8	30.7	39.3	32.9	41.6	32.5	41.9

Table 4: Translation Analysis and Performance of PPI Extraction Procedures

words taken from the English side of the bitext; all language models are built with the SRILM toolkit using Kneser-Ney smoothing (Stolcke, 2002).

The larger system is based on alignments generated over all available C-E bitext (the "ALL C-E" collection of Section 4). The language model is an equal-weight interpolated trigram model trained over 373M English words taken from the English side of the bitext and the LDC Gigaword corpus.

Arabic→English Translation We also evaluate our WtoP alignment models in Arabic-English translation. We report results on a small and a large system. In each, Arabic text is tokenized by the Buckwalter analyzer provided by LDC. We test our models on NIST Arabic/English 2002, 2003 and 2004 (News only) MT evaluation sets that consists of 1043, 663 and 707 Arabic sentences, respectively. Each Arabic sentence has 4 reference translations.

In the small system, the training bitext is from A-E News parallel text, with ∼3.5M words on the English side. We follow the same training procedure and configurations as in Chinese/English system in both translation directions. The language model is an equal-weight interpolated trigram built over ∼400M words from the English side of the bitext, including UN text, and the LDC English Gigaword collection. The large Arabic/English system employs the same language model. Alignments are generated over all A-E bitext available from LDC as of this submission; this consists of approx. 130M words on the English side.

WtoP Model and Model-4 Comparison We first look at translation performance of the small A→E and C→E systems, where alignment models are trained over the smaller bitext collections. The baseline systems (Table 4, line 1) are based on Model-4 Viterbi Phrase-Extract PPIs.

We compare WtoP alignments directly to Model-4 alignments by extracting PPIs from the WtoP alignments using the Viterbi Phrase-Extract procedure (Table 4, line 3). In C→E translation, performance is comparable to that of Model-4; in A→E translation, performance lags slightly. As we add phrase pairs to the WtoP Viterbi Phrase-Extract PPI via the Phrase-Posterior Augmentation procedure (Table 4, lines 4-7), we obtain a ∼1% improvement in BLEU; the value of $T_p = 0.7$ gives improvements across all sets. In C→E translation, this yields good gains relative to Model-4, while in A→E we match or improve the Model-4 performance.

The performance gains through PPI augmentation are consistent with increased PPI coverage of the test set. We tabulate the percentage of test set phrases that appear in each of the PPIs (the 'cvg' values in Table 4). The augmentation scheme is designed specifically to increase coverage, and we find that BLEU score improvements track the phrase coverage of the test set. This is further confirmed by the experiment of Table 4, line 2 in which we take the PPI extracted from Model-4 Viterbi alignments, and add phrase pairs to it using the Phrase-Posterior augmentation scheme with $T_p = 0.7$. We find that the augmentation scheme under the WtoP models can be used to improve the Model-4 PPI itself.

We also investigate C→E and A→E translation performance with PPIs extracted from large bitexts.

Performance of systems based on Model-4 Viterbi Phrase-Extract PPIs is shown in Table 4, line 8. To train Model-4 using GIZA++, we split the bitexts into two (A-E) or three (C-E) partitions, and train models for each division separately; we find that memory usage is otherwise too great. These serve as a single set of alignments for the bitext, as if they had been generated under a single alignment model. When we translate with Viterbi Phrase-Extract PPIs taken from WtoP alignments created over all available bitext, we find comparable performance to the Model-4 baseline (Table 4, line 9). Using the Phrase-Posterior augmentation scheme with $T_p = 0.7$ yields further improvement (Table 4, line 10). Pooling the sets to form two large C→E and A→E test sets, the A→E system improvements are significant at a 95% level (Och, 2003); the C→E systems are only equivalent.

7 Conclusion

We have described word-to-phrase alignment models capable of good quality bitext word alignment. In Arabic-English and Chinese-English translation and alignment they compare well to Model-4, even with large bitexts. The model architecture was inspired by features of Model-4, such as fertility and distortion, but care was taken to ensure that dynamic programming procedures, such as EM and Viterbi alignment, could still be performed. There is practical value in this: training and alignment are easily parallelized. Working with HMMs also makes it straightforward to explore new modeling approaches. We show an augmentation scheme that adds to phrases extracted from Viterbi alignments; this improves translation with both the WtoP and the Model-4 phrase pairs, even though it would be infeasible to implement the scheme under Model-4 itself. We note that these models are still relatively simple, and we anticipate further alignment and translation improvement as the models are refined.

Acknowledgments The TTM translation system was provided by Shankar Kumar. This work was funded by ONR MURI Grant N00014-01-1-0685.

References

A. L. Berger, S. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

P. F. Brown et al. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19:263–312.

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*.

S. Kumar, Y. Deng, and W. Byrne. 2005. A weighted finite state transducer translation template model for statistical machine translation. *Journal of Natural Language Engineering*, 11(3).

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP*.

F. Och and H. Ney. 2000. Improved statistical alignment models. In *Proc. of ACL*, Hong Kong, China.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

F. Och. 2002. *Statistical Machine Translation: From Single Word Models to Alignment Templates*. Ph.D. thesis, RWTH Aachen, Germany.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*.

M. Ostendorf, V. Digalakis, and O. Kimball. 1996. From HMMs to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, 4:360–378.

K. Papineni et al. 2001. BLEU: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. ICSLP*.

E. Sumita et al. 2004. EBMT, SMT, Hybrid and More: ATR spoken language translation system. In *Proc. of the International Workshop on Spoken Language Translation*, Kyoto, Japan.

K. Toutanova, H. T. Ilhan, and C. Manning. 2002. Extentions to HMM-based statistical word alignment models. In *Proc. of EMNLP*.

A. Venugopal, S. Vogel, and A. Waibel. 2003. Effective phrase translation extraction from alignment models. In *Proc. of ACL*.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM based word alignment in statistical translation. In *Proc. of the COLING*.

I. H. Witten and T. C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. In *IEEE Trans. Inform Theory*, volume 37, pages 1085–1094, July.

Inner-Outer Bracket Models for Word Alignment using Hidden Blocks

Bing Zhao
School of Computer Science
Carnegie Mellon University
{bzhao}@cs.cmu.edu

Niyu Ge and **Kishore Papineni**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{niyuge, papineni}@us.ibm.com

Abstract

Most statistical translation systems are based on phrase translation pairs, or "blocks", which are obtained mainly from word alignment. We use blocks to infer better word alignment and improved word alignment which, in turn, leads to better inference of blocks. We propose two new probabilistic models based on the inner-outer segmentations and use EM algorithms for estimating the models' parameters. The first model recovers IBM Model-1 as a special case. Both models outperform bi-directional IBM Model-4 in terms of word alignment accuracy by 10% absolute on the F-measure. Using blocks obtained from the models in actual translation systems yields statistically significant improvements in Chinese-English SMT evaluation.

1 Introduction

Today's statistical machine translation systems rely on high quality phrase translation pairs to acquire state-of-the-art performance, see (Koehn et al., 2003; Zens and Ney, 2004; Och and Ney, 2003). Here, phrase pairs, or "blocks" are obtained automatically from parallel sentence pairs via the underlying word alignments. Word alignments traditionally are based on IBM Models 1-5 (Brown et al., 1993) or on HMMs (Vogel et al., 1996). Automatic word alignment is challenging in that its accuracy is not yet close to inter-annotator agreement in some language pairs: for Chinese-English, inter-annotator agreement exceeds 90 on F-measure whereas IBM Model-4 or HMM accuracy is typically below 80s. HMMs assume that words "close-in-source" are aligned to words "close-in-target". While this locality assumption is generally sound, HMMs do have limitations: the self-transition probability of a state (word) is the only control on the duration in the state, the length of the phrase aligned to the word. Also there is no natural way to control repeated non-contiguous visits to a state. Despite these problems, HMMs remain attractive for their speed and reasonable accuracy.

We propose a new method for localizing word alignments. We use blocks to achieve locality in the following manner: a block in a sentence pair is a source phrase aligned to a target phrase. We assume that words inside the source phrase cannot align to words outside the target phrase and that words outside the source phrase cannot align to words inside the target phrase. Furthermore, a block divides the sentence pair into two smaller regions: the *inner* part of the block, which corresponds to the source and target phrase in the block, and the *outer* part of the block, which corresponds to the remaining source and target words in the parallel sentence pair. The two regions are non-overlapping; and each of them is shorter than the original parallel sentence pair. The regions are thus easier to align than the original sentence pairs (e.g., using IBM Model-1). While the model uses a single block to split the sentence pair into two independent regions, it is not clear which block we should select for this purpose. Therefore, we treat the splitting block as a hidden variable.

This proposed approach is far simpler than treating the entire sentence as a sequence of non-overlapping phrases (or chunks) and considering such sequential segmentation either explicitly or implicitly. For example, (Marcu and Wong, 2002) for a joint phrase based model, (Huang et al., 2003) for a translation memory system; and (Watanabe et al., 2003) for a complex model of insertion, deletion and head-word driven chunk reordering. Other approaches including (Watanabe et al., 2002) treat extracted phrase-pairs as new parallel data with limited success. Typically, they share a similar architecture of phrase level segmentation, reordering, translation as in (Och and Ney, 2002; Koehn and Knight, 2002; Yamada and Knight, 2001). The phrase level interaction has to be taken care of for the non-overlapping sequential segmentation in a complicated way. Our models model such interactions in a soft way. The hidden blocks are allowed to overlap with each other, while each block induced two non-overlapping regions, i.e. the model brackets the sentence pair into two independent parts which are generated synchronously. In this respect, it resembles bilingual bracketing (Wu, 1997), but our model has more lexical items in the blocks with many-to-many word alignment freedom in both inner and outer parts.

We present our localization constraints using blocks for word alignment in Section 2; we detail our two new probabilistic models and their EM training algorithms in Section 3; our baseline system, a maximum-posterior inference for word alignment, is explained in Section 4; experimental results of alignments and translations are in Section 5; and Section 6 contains discussion and conclusions.

2 Segmentation by a Block

We use the following notation in the remainder of this paper: $\mathbf{e}$ and $\mathbf{f}$ denote the English and foreign sentences with sentence lengthes of I and J, respectively. e_i is an English word at position i in $\mathbf{e}$; f_j is a foreign word at position j in $\mathbf{f}$. $\mathbf{a}$ is the alignment vector with a_j mapping the position of the English word e_{a_j} to which f_j connects. Therefore, we have the standard limitation that one foreign word cannot be connected to more than one English word. A *block* $\delta^{\square}$ is defined as a pair of brackets as follows:

$$\delta^{\square} = (\delta^{\mathbf{e}}, \delta^{\mathbf{f}}) = ([i_l, i_r], [j_l, j_r]), \tag{1}$$

where $\delta^{\mathbf{e}} = [i_l, i_r]$ is a bracket in English sentence defined by a pair of indices: the *left* position i_l and the *right* position i_r, corresponding to a English phrase $e_{i_l}^{i_r}$. Similar notations are for $\delta^{\mathbf{f}} = [j_l, j_r]$, which is one possible *projection* of $\delta^{\mathbf{e}}$ in $\mathbf{f}$. The subscript l and r are abbreviations of left and right, respectively.

$\delta^{\mathbf{e}}$ segments $\mathbf{e}$ into two parts: $(\delta^{\mathbf{e}}, \mathbf{e}) = (\delta^{\mathbf{e}}_{\in}, \delta^{\mathbf{e}}_{\notin})$. The inner part $\delta^{\mathbf{e}}_{\in} = \{e_i, i \in [i_l, i_r]\}$ and the outer part $\delta^{\mathbf{e}}_{\notin} = \{e_i, i \notin [i_l, i_r]\}$; $\delta^{\mathbf{f}}$ segments $\mathbf{f}$ similarly.

Thus, the block $\delta^{\square}$ splits the parallel sentence pair into two *non-overlapping* regions: the *Inner* $\delta^{\square}_{\in}$ and *Outer* $\delta^{\square}_{\notin}$ parts (see Figure 1). With this segmentation, we assume the words in the inner part are aligned to inner part only: $\delta^{\square}_{\in} = \delta^{\mathbf{e}}_{\in} \leftrightarrow \delta^{\mathbf{f}}_{\in} : \{e_i, i \in [i_l, i_r]\} \leftrightarrow \{f_j, j \in [j_l, j_r]\}$; and words in the outer part are aligned to outer part only: $\delta^{\square}_{\notin} = \delta^{\mathbf{e}}_{\notin} \leftrightarrow \delta^{\mathbf{f}}_{\notin} : \{e_i, i \notin [i_l, i_r]\} \leftrightarrow \{f_j, j \notin [j_l, j_r]\}$. We do not allow alignments to cross block boundaries. Words inside a block $\delta^{\square}$ can be aligned using a variety of models (IBM models 1-5, HMM, etc). We choose Model1 for simplicity. If the block boundaries are accurate, we can expect high quality word alignment. This is our proposed new localization method.

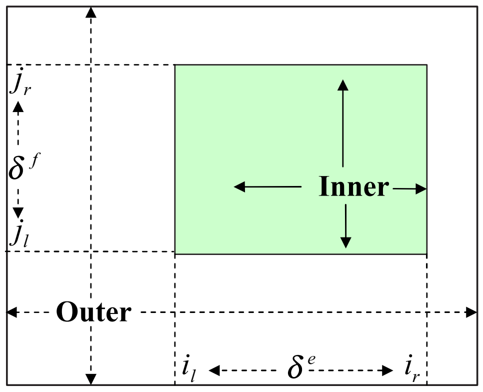


Figure 1: Segmentation by a Block

3 Inner-Outer Bracket Models

We treat the constraining block as a hidden variable in a generative model shown in Eqn. 2.

$$\begin{aligned} P(\mathbf{f}|\mathbf{e}) &= \sum_{\{\delta^{\square}\}} P(\mathbf{f}, \delta^{\square}|\mathbf{e}) \\ &= \sum_{\{\delta^{\mathbf{e}}\}} \sum_{\{\delta^{\mathbf{f}}\}} P(\mathbf{f}, \delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e}) P(\delta^{\mathbf{e}}|\mathbf{e}), \end{aligned} \tag{2}$$

where $\delta^{\square} = (\delta^{\mathbf{e}}, \delta^{\mathbf{f}})$ is the hidden block. In the generative process, the model first generates a bracket $\delta^{\mathbf{e}}$ for $\mathbf{e}$ with a monolingual bracketing model of $P(\delta^{\mathbf{e}}|\mathbf{e})$. It then uses the segmentation of the English $(\delta^{\mathbf{e}}, \mathbf{e})$ to generate the projected bracket $\delta^{\mathbf{f}}$ of $\mathbf{f}$ using a generative translation model $P(\mathbf{f}, \delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e}) = P(\delta^{\mathbf{f}}_{\notin}, \delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\notin}, \delta^{\mathbf{e}}_{\in})$ — the key model to implement our proposed inner-outer constraints. With the hidden block $\delta^{\square}$ inferred, the model then generates word alignments within the inner and outer parts separately. We present two generating processes for the inner and outer parts induced by $\delta^{\square}$ and corresponding two models of $P(\mathbf{f}, \delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e})$. These models are described in the following secions.

3.1 Inner-Outer Bracket Model-A

The first model assumes that the inner part and the outer part are generated independently. By the formal equivalence of (f, δ^f) with $(\delta^f_{\in}, \delta^f_{\notin})$, Eqn. 2 can be approximated as:

$$P(\mathbf{f}|\mathbf{e}) \approx \sum_{\{\delta^{\mathbf{e}}\}} \sum_{\{\delta^{\mathbf{f}}\}} P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in}) P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin}) P(\delta^{\mathbf{e}}|\mathbf{e}) P(\delta^{\mathbf{f}}|\delta^{\mathbf{e}}), \tag{3}$$

where $P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})$ and $P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin})$ are two independent generative models for inner and outer parts, respectively and are futher decompsed into:

$$P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in}) = \sum_{\{a_j \in \delta^{\mathbf{e}}_{\in}\}} \prod_{f_j \in \delta^{\mathbf{f}}_{\in}} P(f_j|e_{a_j}) P(e_{a_j}|\delta^{\mathbf{e}}_{\in})$$

$$P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin}) = \sum_{\{a_j \in \delta^{\mathbf{e}}_{\notin}\}} \prod_{f_j \in \delta^{\mathbf{f}}_{\notin}} P(f_j|e_{a_j}) P(e_{a_j}|\delta^{\mathbf{e}}_{\notin}), \quad (4)$$

where $\{a_1^J\}$ is the word alignment vector. Given the block segmentation and word alignment, the generative process first randomly selects a e_i according to either $P(e_i|\delta^{\mathbf{e}}_{\in})$ or $P(e_i|\delta^{\mathbf{e}}_{\notin})$; and then generates f_j indexed by word alignment a_j with $i = a_j$ according to a word level lexicon $P(f_j|e_{a_j})$. This generative process using the two models of $P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})$ and $P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin})$ must satisfy the constraints of segmentations induced by the hidden block $\delta^{\square} = (\delta^{\mathbf{e}}, \delta^{\mathbf{f}})$. The English words $\delta^{\mathbf{e}}_{\in}$ inside the block can only generate the words in $\delta^{\mathbf{f}}_{\in}$ and nothing else; likewise $\delta^{\mathbf{e}}_{\notin}$ only generates $\delta^{\mathbf{f}}_{\notin}$. Overall, the combination of $P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin})$ in Eqn. 3 collaborates each other quite well in practice. For a particular observation $\delta^{\mathbf{f}}_{\in}$, if $\delta^{\mathbf{e}}_{\in}$ is too small (i.e., missing translations), $P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})$ will suffer; and if $\delta^{\mathbf{e}}_{\in}$ is too big (i.e., robbing useful words from $\delta^{\mathbf{e}}_{\notin}$), $P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin})$ will suffer. Therefore, our proposed model in Eqn. 3 combines the two costs and requires both inner and outer parts to be explained well at the same time.

Because the model in Eqn. 3 is essentially a two-level ($\delta^{\square}$ and $\mathbf{a}$) mixture model similar to IBM Models, the EM algorithm is quite straight forward as in IBM models. Shown in the following are several key E-step computations of the posteriors. The M-step (optimization) is simply the normalization of the fractional counts collected using the posteriors through the inference results from E-step:

$$P_{\delta^{\square}_{\in}}(a_j|\delta^{\mathbf{f}}_{\in}, \delta^{\mathbf{e}}_{\in}) = \frac{P(f_j|e_{a_j})}{\sum_{e_k \in \delta^{\mathbf{e}}_{\in}} P(f_j|e_k)}$$

$$P_{\delta^{\square}_{\notin}}(a_j|\delta^{\mathbf{f}}_{\notin}, \delta^{\mathbf{e}}_{\notin}) = \frac{P(f_j|e_{a_j})}{\sum_{e_k \in \delta^{\mathbf{e}}_{\notin}} P(f_j|e_k)} \quad (5)$$

The posterior probability of $P(a_1^J|\mathbf{f}, \delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e}) = \prod_{j=1}^{J} P(a_j|\mathbf{f}, \delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})$, where $P(a_j|\mathbf{f}, \delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})$ is either $P_{\delta^{\square}_{\in}}(a_j|\delta^{\mathbf{f}}_{\in}, \delta^{\mathbf{e}}_{\in})$ when $(f_j, e_{a_j}) \in \delta^{\square}_{\in}$, or otherwise $P_{\delta^{\square}_{\notin}}(a_j|\delta^{\mathbf{f}}_{\notin}, \delta^{\mathbf{e}}_{\notin})$ when $(f_j, e_{a_j}) \in \delta^{\square}_{\notin}$. Assuming $P(\delta^{\mathbf{e}}|\mathbf{e})$ to be a uniform distribution, the posterior of selecting a hidden block given observations: $P(\delta^{\square} = (\delta^{\mathbf{e}}, \delta^{\mathbf{f}})|\mathbf{e}, \mathbf{f})$ is proportional to block level relative frequency $P_{rel}(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})$ updated in each iteration; and can be smoothed with $P(\delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{f}, \mathbf{e}) = P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin}) / \sum_{\{\delta'^{\mathbf{f}}\}} P(\delta'^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in})P(\delta'^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin})$ assuming Model-1 alignment in the inner and outer parts independently to reduce the risks of data sparseness in estimations.

In principle, $\delta^{\mathbf{e}}$ can be a bracket of any length not exceeding the sentence length. If we restrict the bracket length to that of the sentence length, we recover IBM Model-1. Figure 2 summarizes the generation process for Inner-Outer Bracket Model-A.

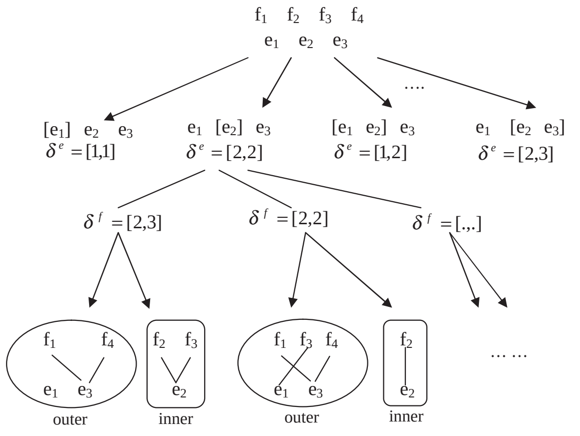


Figure 2: Illustration of generative Bracket Model-A

3.2 Inner-Outer Bracket Model-B

A block $\delta^{\square}$ invokes both the inner and outer generations simultaneously in Bracket Model A (BM-A). However, the generative process is usually more effective in the inner part as $\delta^{\square}$ is generally small and accurate. We can build a model focusing on generating only the inner part with careful inferences to avoid errors from noisy blocks. To ensure that all f_1^J are generated, we need to propose enough blocks to cover each observation f_j. This constraint can be met by treating the whole sentence pair as one block.

The generative process is as follows: First the model generates an English bracket $\delta^{\mathbf{e}}$ as before. The model then generates a projection $\delta^{\mathbf{f}}$ in $\mathbf{f}$ to localize all a_j's for the given $\delta^{\mathbf{e}}$ according to $P(\delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e})$. $\delta^{\mathbf{e}}$ and $\delta^{\mathbf{f}}$ forms a hidden block $\delta^{\square}$. Given $\delta^{\square}$, the model then generates only the inner part $f_j \in \delta^{\mathbf{f}}_{\in}$ via $P(\mathbf{f}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e}) \simeq P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})$. Eqn. 6 summarizes this by rewriting $P(\mathbf{f}, \delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e})$:

$$\begin{aligned} P(\mathbf{f}, \delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e}) &= P(\mathbf{f}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})P(\delta^{\mathbf{f}}|\delta^{\mathbf{e}}, \mathbf{e}) \quad (6) \\ &= P(\mathbf{f}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})P([j_l, j_r]|\delta^{\mathbf{e}}, \mathbf{e}) \\ &\simeq P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})P([j_l, j_r]|\delta^{\mathbf{e}}, \mathbf{e}). \end{aligned}$$

$P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e})$ is a bracket level *emission* probabilistic model which generates a bag of contiguous words $f_j \in \delta^{\mathbf{f}}_{\in}$ under the constraints from the given hidden block $\delta^{\square} = (\delta^{\mathbf{f}}, \delta^{\mathbf{e}})$. The model is simplified in Eqn. 7 with the assumption of bag-of-words' independence within the bracket $\delta^{\mathbf{f}}$:

$$P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e}) = \sum_{a_1^J} \prod_{j \in \delta^{\mathbf{f}}_{\in}} P(f_j|e_{a_j}) P(e_{a_j}|\delta^{\mathbf{f}}, \delta^{\mathbf{e}}, \mathbf{e}). \quad (7)$$

The $P([j_l, j_r]|\delta^{\mathbf{e}}, \mathbf{e})$ in Eqn. 6 is a *localization* probabilistic model, which has resemblances to an HMM's transition probability, $P(a_j|a_{j-1})$, implementing the assumption "close-in-source" is aligned to "close-in-target". However, instead of using the simple position variable a_j, $P([j_l, j_r]|\delta^{\mathbf{e}}, \mathbf{e})$ is more expressive with word identities to localize words $\{f_j\}$ aligned to $\delta^{\mathbf{e}}_{\in}$. To model $P([j_l, j_r]|\delta^{\mathbf{e}}, \mathbf{e})$ reliably, $\delta^{\mathbf{f}} = [j_l, j_r]$ is equivalently defined as the *center* and *width* of the bracket $\delta^{\mathbf{f}}$: $(\odot_{\delta^{\mathbf{f}}}, w_{\delta^{\mathbf{f}}})$. To simplify it further, we assume that $w_{\delta^{\mathbf{f}}}$ and $\odot_{\delta^{\mathbf{f}}}$ can be predicted independently.

The *width* model, $P(w_{\delta^{\mathbf{f}}}|\delta^{\mathbf{e}}, \mathbf{e})$, depends on the length of the English bracket and the fertilities of English words in it. To simplify M-step computations, we can compute the expected width as in Eqn. 8.

$$E\{w_{\delta^{\mathbf{f}}}|\delta^{\mathbf{e}}, \mathbf{e}\} \simeq \gamma \cdot |i_r - i_l + 1|, \tag{8}$$

where γ is the expected bracket length ratio and is approximated by the average sentence length ratio computed using the whole parallel corpus. For Chinese-English, $\gamma = 1/1.3 = 0.77$. In practice, this estimation is quite reliable.

The *center* model $P(\odot_{\delta^{\mathbf{f}}}|\delta^{\mathbf{e}}, \mathbf{e})$ is harder. It is conditioned on the translational equivalence between the English bracket and its projection. We compute the expected $\odot_{\delta^{\mathbf{f}}}$ by averaging the weighted expected centers from all the English words in $\delta^{\mathbf{e}}$ as in Eqn. 9.

$$\begin{aligned} E\{\odot_{\delta^{\mathbf{f}}}|\delta^{\mathbf{e}}, \mathbf{e}\} &= \textstyle\sum_{j=0}^{J} j \cdot P(j|\delta^{\mathbf{e}}, \mathbf{e}) \\ &\simeq \textstyle\sum_{j=0}^{J} j \cdot \frac{\sum_{i \in \delta^{\mathbf{e}}} P(f_j|e_i)}{\sum_{j'=0}^{J} \sum_{i \in \delta^{\mathbf{e}}} P(f_{j'}|e_i)}. \end{aligned} \tag{9}$$

The expectations of $(\odot_{\delta^{\mathbf{f}}}, w_{\delta^{\mathbf{f}}})$ from Eqn. 8 and Eqn. 9 give a reliable starting point for a local search for the optimal estimation of $(\hat{\odot}_{\delta^{\mathbf{f}}}, \hat{w}_{\delta^{\mathbf{f}}})$ as in Eqn 10:

$$(\hat{\odot}_{\delta^{\mathbf{f}}}, \hat{w}_{\delta^{\mathbf{f}}}) = \underset{\{(\odot_{\delta^{\mathbf{f}}}, w_{\delta^{\mathbf{f}}})\}}{\arg\max}\ P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in}) P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin}), \tag{10}$$

where the score functions of $P(\delta^{\mathbf{f}}_{\in}|\delta^{\mathbf{e}}_{\in}) P(\delta^{\mathbf{f}}_{\notin}|\delta^{\mathbf{e}}_{\notin})$ are in Eqn. 4 with the word alignment explicitly given from the previous iteration. For the very first iteration, *full alignment* is assumed; this means that every word pair is connected in the parallel sentences. During the local search in Eqn. 10, one can choose the top-1 (Viterbi) $(\hat{\odot}_{\delta^{\mathbf{f}}}, \hat{w}_{\delta^{\mathbf{f}}})$ or top-N candidates and normalize over these candidates to obtain the posteriors. Except for the local search of $(\hat{\odot}_{\delta^{\mathbf{f}}}, \hat{w}_{\delta^{\mathbf{f}}})$, the remainder EM steps are similar to Bracket Model-A, though with different interpretations.

By performing local search in Eqn. 10, Model-B localizes hidden blocks more accurately than the scheme of the smoothed relative frequency in Model-A's EM iterations. The model is also more focused on the predictions in the inner part. Figure 3 summarizes the generative process of Model-B (BM-B).

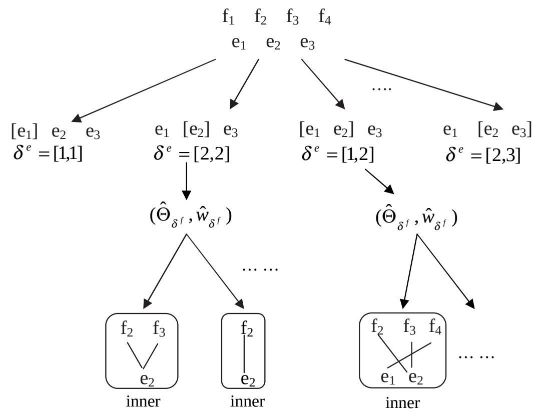


Figure 3: Generative Bracket Model-B

3.3 A Null Word Model

The null word model allows words to be aligned to nothing. In the traditional IBM models, there is a universal null word which is attached to every sentence pair to compete with word generators. In our inner-outer bracket models, we use two context-specific null word models which use both the left and right context as competitors in the generative process for each observation f_j: $P(f_j|f_{j-1}, \mathbf{e})$ and $P(f_j|f_{j+1}, \mathbf{e})$. This is similar to the approach in (Toutanova et al., 2002), in which the null word model is part of an extended HMM using left context only. With two null word models, we can associate f_j with its left or right context (i.e., a null link) when the null word models are very strong, or when the word's alignment is too far from the expected center $\hat{\odot}_{\delta^{\mathbf{f}}}$ in Eqn. 9.

4 A Max-Posterior for Word Alignment

In the HMM framework, (Ge, 2004) proposed a *maximum-posterior* method which worked much better than *Viterbi* for Arabic to English translations. The difference between maximum-posterior and Viterbi, in a nutshell, is that while Viterbi computes the best state *sequence* given the observation, the maximum-posterior computes the best state *one at a time.*

In the terminology of HMM, let the states be the words in the foreign sentence f_1^J and observations be the words in the English sentence e_1^T. We use the subscript t to note the fact that e_t is observed (or emitted) at time step t. The posterior probabilities $P(f_j|e_t)$ (state given observation) are obtained after the forward-backward training. The maximum-posterior word alignments are obtained by first computing a pair $(j,t)^*$:

$$(j,t)^* = \arg\max_{(j,t)} P(f_j|e_t), \qquad (11)$$

that is, the point at which the posterior is maximum. The pair (j,t) defines a word pair (f_j, e_t) which is then aligned. The procedure continues to find the next maximum in the posterior matrix. Contrast this with Viterbi alignment where one computes

$$\hat{f}_1^T = \arg\max_{\{f_1^T\}} P(f_1, f_2, \cdots, f_T|e_1^T), \qquad (12)$$

We observe, in parallel corpora, that when one word translates into multiple words in another language, it usually translates into a contiguous sequence of words. Therefore, we impose a contiguity constraint on word alignments. When one word f_j aligns to multiple English words, the English words must be contiguous in $\mathbf{e}$ and vice versa. The algorithm to find word alignments using max-posterior with contiguity constraint is illustrated in Algorithm 1.

Algorithm 1 A maximum-posterior algorithm with contiguity constraint

```
1: while $(j,t) = (j,t)^*$ (as computed in Eqn. 11) do
2:   if $(f_j, e_t)$ is not yet aligned then
3:     align($f_j$, $e_t$);
4:   else if ($e_t$ is contiguous to what $f_j$ is aligned) or ($f_j$ is contiguous to what $e_t$ is aligned) then
5:     align($f_j$, $e_t$);
6:   end if
7: end while
```

The algorithm terminates when there isn't any 'next' posterior maximum to be found. By definition, there are at most JxT 'next' maximums in the posterior matrix. And because of the contiguity constraint, not all (f_j, e_t) pairs are valid alignments. The algorithm is sure to terminate. The algorithm is, in a sense, *directionless*, for one f_j can align to multiple e_t's and vise versa as long as the multiple connections are contiguous. Viterbi, however, is *directional* in which one state can emit multiple observations but one observation can only come from one state.

5 Experiments

We evaluate the performances of our proposed models in terms of word alignment accuracy and translation quality. For word alignment, we have 260 hand-aligned sentence pairs with a total of 4676 word pair links. The 260 sentence pairs are randomly selected from the CTTP[1] corpus. They were then word aligned by eight bilingual speakers. In this set, we have one-to-one, one-to-many and many-to-many alignment links. If a link has one target *functional* word, it is considered to be a *functional* link (Examples of funbctional words are prepositions, determiners, etc. There are in total 87 such functional words in our experiments). We report the overall F-measures as well as F-measures for both content and functional word links. Our significance test shows an overall interval of ±1.56% F-measure at a 95% confidence level.

For training data, the small training set has 5000 sentence pairs selected from XinHua news stories with a total of 131K English words and 125K Chinese words. The large training set has 181K sentence pairs (5k+176K); and the additional 176K sentence pairs are from FBIS and Sinorama, which has in total 6.7 million English words and 5.8 million Chinese words.

5.1 Baseline Systems

The baseline is our implementation of HMM with the maximum-posterior algorithm introduced in section 4. The HMMs are trained unidirectionally. IBM Model-4 is trained with GIZA++ using the best reported settings in (Och and Ney, 2003). A few parameters, especially the maximum fertility, are tuned for GIZA++'s optimal performance. We collect *bi-directional* (**bi**) refined word alignment by growing the intersection of *Chinese-to-English* (**CE**) alignments and *English-to-Chinese* (**EC**) alignments with the neighboring unaligned word pairs which appear in the union similar to the "final-and" approaches (Koehn, 2003; Och and Ney, 2003; Tillmann, 2003). Table 1 summarizes our baseline with different settings. Table 1 shows that **HMM EC-P** gives the

F-measure(%)		**Func**	**Cont**	**Both**
Small	**HMM EC-P**	**54.69**	**69.99**	**64.78**
	HMM EC-V	31.38	53.56	55.59
	HMM CE-P	51.44	69.35	62.69
	HMM CE-V	31.43	63.84	55.45
Large	**HMM EC-P**	**60.08**	78.01	**71.92**
	HMM EC-V	32.80	74.10	64.26
	HMM CE-P	58.45	**79.44**	71.84
	HMM CE-V	35.41	79.12	68.33
Small	GIZA MH-bi	45.63	69.48	60.08
	GIZA M4-bi	48.80	73.68	63.75
Large	GIZA MH-bi	49.13	76.51	65.67
	GIZA M4-bi	52.88	81.76	70.24
-	Fully-Align [2]	5.10	15.84	9.28

Table 1: Baseline: **V**: **V**iterbi; **P**: Max-**P**osterior

[1]LDC2002E17

best baseline, better than bidirectional refined word alignments from GIZA M4 and the HMM Viterbi aligners.

5.2 Inner-Outer Bracket Models

We trained HMM lexicon $P(f|e)$ to initialize the inner-outer Bracket models. Afterwards, up to 15–20 EM iterations are carried out. Iteration starts from the fully aligned[2] sentence pairs, which give an F-measure of 9.28% at iteration one.

5.2.1 Small Data Track

Figure 4 shows the performance of Model-A (BM-A) trained on the small data set. For each English bracket, *Top-1* means only the fractional counts from the Top-1 projection are collected, *Top-all* means counts from all possible projections are collected. *Inside* means the fractional counts are collected from the inner part of the block only; and *outside* means they are collected from the outer parts only. Using the Top-1 projection from the inner parts of the block (*top-1-inside*) gives the best performance: an F-measure of 72.29%, or a 7.5% absolute improvement over the best baseline at iteration 5. Figure 5 shows

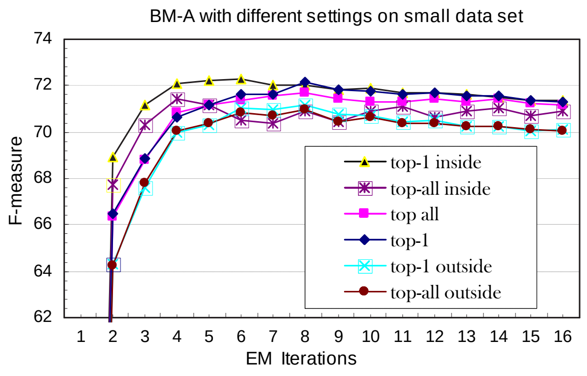


Figure 4: BM-A with different settings on small data

the performance of Inner-Outer Bracket Model-B (BM-B) over EM iterations. *smoothing* means when collecting the fractional counts, we reweigh the updated fractional count by 0.95 and give the remaining 0.05 weight to original fractional count from the links, which were aligned in the previous iteration. *w/null* means we applied the proposed Null word model in section 3.3 to infer null links. We also predefined a list of 15 English function words, for which there might be no corresponding Chinese words as translations. These 15 English words are "*a, an, the, of, to, for, by, up, be, been, being, does, do, did, -*". In the *drop-null* experiments, the links containing these predefined function words are simply dropped in the final word alignment (this means they are left unaligned).

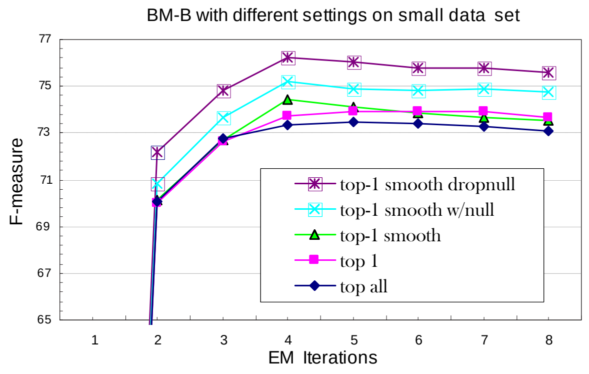


Figure 5: BM-B with different settings on small data

Empirically we found that doing more than 5 iterations lead to overfitting. The peak performance in our model is usually achieved around iteration 4∼5. At iteration 5, setting "BM-B Top-1" gives an F-measure of 73.93% which is better than BM-A's best performance (72.29%). This is because Model B leverages a local search for less noisy blocks and hence the inner part is more accurately generated (which in turn means the outer part is also more accurate). From this point on, all of our experiments are using Model B. With smoothing, BM-B improves to 74.46%. After applying the null word model, we get 75.20%. By simply dropping links containing the 15 English functional words, we get 76.24%, which is significantly better than our best baseline obtained from even the large training set (HMM EC-P: 71.92%).

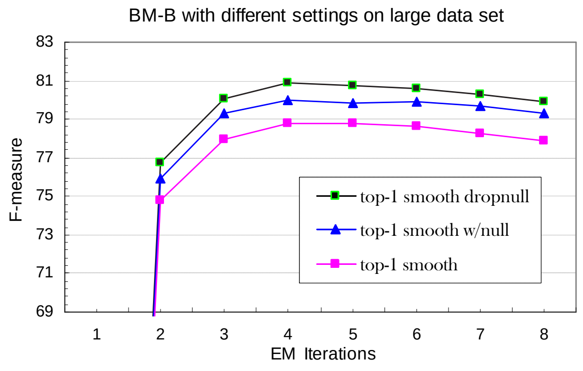


Figure 6: BM-B with different settings on large data

5.2.2 Large Data Track

Figure 6 shows performance pictures of model BM-B on the large training set. Without dropping English functional words, the best performance is

[2]Every possible word pair is aligned

80.38% at iteration 4 using the Top-1 projection together with the null word models. By additionally dropping the links containing the 15 functional English words, we get 81.47%. These results are all significantly better than our strongest baseline system: 71.92% F-measure using HMM EC-P (70.24% using bidirectional Model-4 for comparisons).

On this data set, we experimented with different maximum bracket length limits, from one word (unigram) to nine-gram. Results show that a maximum bracket length of four is already optimal (79.3% with top-1 projection), increased from 62.4% when maximum length is limited to one. No improvements are observed using longer than five-gram.

5.3 Evaluate Blocks in the EM Iterations

Our intuition was that good blocks can improve word alignment and, in turn, good word alignment can lead to better block selection. The experimental results above support the first claim. Now we consider the second claim that good word alignment leads to better block selection.

Given reference human word alignment, we extract reference blocks up to five-gram phrases on Chinese. The block extraction procedure is based on the procedures in (Tillmann, 2003).

During EM, we output all the hidden blocks actually inferred at each iteration, then we evaluate the precision, recall and F-measure of the hidden blocks according to the extracted reference blocks. The results are shown in Figure 7. Because we extract all

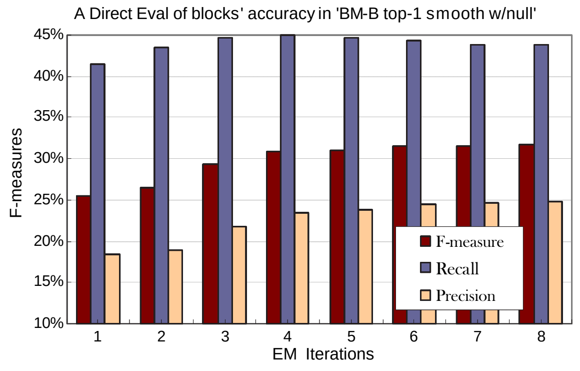


Figure 7: A Direct Eval. of Blocks in BM-B

possible n-grams at each position in **e**, the precision is low and the recall is relatively high as shown by Figure 7. It also shows that blocks do improve, presumably benefiting from better word alignments.

Table 2 summarizes word alignment performances of Inner-Outer BM-B in different settings. Overall, without the handcrafted function word list, BM-B gives about 8% absolute improvement in F-measure on the large training set and 9% for the small set with a confidence interval of ±1.56%.

F-measure(%)		**Func**	**Cont**	**Both**
Small	Baseline	54.69	69.99	64.78
	BM-B-drop	**62.76**	**82.99**	**76.24**
	BM-B w/null	61.24	82.54	75.19
	BM-B smooth	59.61	82.99	74.46
Large	Baseline	60.08	78.01	71.92
	BM-B-drop	**63.95**	**90.09**	**81.47**
	BM-B w/null	62.24	89.99	80.38
	BM-B smooth	60.49	90.09	79.31

Table 2: BM-B with different settings

5.4 Translation Quality Evaluations

We also carried out the translation experiments using the best settings for Inner-Outer BM-B (i.e. *BM-B-drop*) on the TIDES Chinese-English 2003 test set. We trained our models on 354,252 test-specific sentence pairs drawn from LDC-supplied parallel corpora. On this training data, we ran 5 iterations of EM using BM-B to infer word alignments. A monotone decoder similar to (Tillmann and Ney, 2003) with a trigram language model[3] is set up for translations. We report *case sensitive Bleu* (Papineni et al., 2002) score **BleuC** for all experiments. The baseline system (*HMM*) used phrase pairs built from the HMM-EC-P maximum posterior word alignment and the corresponding lexicons. The baseline BleuC score is 0.2276 ± 0.015. If we use the phrase pairs built from the bracket model instead (but keep the HMM trained lexicons), we get case sensitive BleuC score 0.2526. The improvement is statistically significant. If on the other hand, we use baseline phrase pairs with bracket model lexicons, we get a BleuC score 0.2325, which is only a marginal improvement. If we use *both* phrase pairs and lexicons from the bracket model, we get a case sensitive BleuC score 0.2750, which is a statistically significant improvement. The results are summarized in Table 3.

Settings	**BleuC**
Baseline (HMM phrases and lexicon)	0.2276
Bracket phrases and HMM lexicon	0.2526
Bracket lexicon and HMM phrases	0.2325
Bracket (phrases and lexicon)	0.2750

Table 3: Improved *case sensitive* BleuC using BM-B

Overall, using Model-B, we improve translation quality from 0.2276 to 0.2750 in case sensitive BleuC score.

[3]Trained on 1-billion-word ViaVoice English data; the same data is used to build our True Caser.

6 Conclusion

Our main contributions are two novel Inner-Outer Bracket models based on segmentations induced by hidden blocks. Modeling the Inner-Outer hidden segmentations, we get significantly improved word alignments for both the small training set and the large training set over the widely-practiced bidirectional IBM Model-4 alignment. We also show significant improvements in translation quality using our proposed bracket models. Robustness to noisy blocks merits further investigation.

7 Acknowledgement

This work is supported by DARPA under contract number N66001-99-28916.

References

P.F. Brown, Stephen A. Della Pietra, Vincent. J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics*, volume 19(2), pages 263–331.

Niyu Ge. 2004. A maximum posterior method for word alignment. In *Presentation given at DARPA/TIDES MT workshop.*

J.X. Huang, W. Wang, and M. Zhou. 2003. A unified statistical model for generalized translation memory system. In *Machine Translation Summit IX*, pages 173–180, New Orleans, USA, September 23-27.

Philipp Koehn and Kevin Knight. 2002. Chunkmt: Statistical machine translation with richer linguistic knowledge. Draft, Unpublished.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based machine translation. In *Proc. of HLT-NAACL 2003*, pages 48–54, Edmonton, Canada, May-June.

Philipp Koehn. 2003. Noun phrase translation. In *Ph.D. Thesis*, University of Southern California, ISI.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 133–139, Philadelphia, PA, July 6-7.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of ACL*, pages 440–447.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 29, pages 19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Conf. of the ACL (ACL 02)*, pages 311–318, Philadelphia, PA, July.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dp beam search algorithm for statistical machine translation. In *Computational Linguistics*, volume 29(1), pages 97–133.

Christoph Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing.*

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to hmm-based statistical word alignment models. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, July 6-7.

S. Vogel, Hermann Ney, and C. Tillmann. 1996. Hmm based word alignment in statistical machine translation. In *Proc. The 16th Int. Conf. on Computational Lingustics, (Coling'96)*, pages 836–841, Copenhagen, Denmark.

Taro Watanabe, Kenji Imamura, and Eiichiro Sumita. 2002. Statistical machine translation based on hierarchical phrases. In *9th International Conference on Theoretical and Methodological Issues*, pages 188–198, Keihanna, Japan, March.

Taro Watanabe, Eiichiro Sumita, and Hiroshi G. Okuno. 2003. Chunk-based statistical translation. In *In 41st Annual Meeting of the ACL (ACL 2003)*, pages 303–310, Sapporo, Japan.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In *Computational Linguistics*, volume 23(3), pages 377–403.

K. Yamada and Kevin. Knight. 2001. Syntax-based statistical translation model. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL-2001).*

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of the Human Language Technology Conference (HLT-NAACL)s*, pages 257–264, Boston, MA, May.

Alignment Link Projection Using Transformation-Based Learning

Necip Fazil Ayan, Bonnie J. Dorr and **Christof Monz**
Department of Computer Science
University of Maryland
College Park, MD 20742
{nfa,bonnie,christof}@umiacs.umd.edu

Abstract

We present a new word-alignment approach that learns errors made by existing word alignment systems and corrects them. By adapting transformation-based learning to the problem of word alignment, we project new alignment links from already existing links, using features such as POS tags. We show that our alignment link projection approach yields a significantly lower alignment error rate than that of the best performing alignment system (22.6% relative reduction on English-Spanish data and 23.2% relative reduction on English-Chinese data).

1 Introduction

Word-level alignment is a critical component of a wide range of NLP applications, such as construction of bilingual lexicons (Melamed, 2000), word sense disambiguation (Diab and Resnik, 2002), projection of language resources (Yarowsky et al., 2001), and statistical machine translation. Although word-level aligners tend to perform well when there is *enough* training data, the quality of word alignment decreases as the size of training data decreases. Moreover, word-alignment systems are often tripped up by many-to-many correspondences, morphological language distinctions, paraphrased and free translations, and a high percentage of function words (about 50% of the tokens in most texts).

At the heart of the matter is a set of assumptions that word-alignment algorithms must make in order to reduce the hypothesis space, since word alignment is an exponential problem. Because of these assumptions, learning algorithms tend to make similar errors throughout the entire data.

This paper presents a new approach—*Alignment Link Projection (ALP)*—that learns common alignment errors made by an alignment system and attempts to correct them. Our approach assumes the initial alignment system adequately captures certain kinds of word correspondences but fails to handle others. ALP starts with an initial alignment and then fills out (i.e., *projects*) new word-level alignment relations (i.e., *links*) from existing alignment relations. ALP then deletes certain alignment links associated with common errors, thus improving precision and recall.

In our approach, we adapt transformation-based learning (TBL) (Brill, 1995; Brill, 1996) to the problem of word alignment. ALP attempts to find an ordered list of transformation rules (within a prespecified search space) to improve a baseline annotation. The rules decompose the search space into a set of consecutive words (windows) within which alignment links are added, to or deleted from, the initial alignment. This window-based approach exploits the clustering tendency of alignment links, i.e., when there is a link between two words, there is frequently another link in close proximity.

TBL is an appropriate choice for this problem for the following reasons:

1. It can be optimized directly with respect to an evaluation metric.
2. It learns rules that improve the initial prediction iteratively, so that it is capable of correcting previous errors in subsequent iterations.
3. It provides a readable description (or classification) of errors made by the initial system, thereby enabling alignment refinements.

The rest of the paper is organized as follows: In the next section we describe previous work on improving word alignments. Section 3 presents a brief overview of TBL. Section 4 describes the adaptation of TBL to the word alignment problem. Section 5 compares ALP to various alignments and presents results on English-Spanish and English-Chinese. We show that ALP yields a significant reductions in alignment error rate over that of the best performing alignment system.

2 Related Work

One of the major problems with the IBM models (Brown et al., 1993) and the HMM models (Vogel et al., 1996) is that they are restricted to the alignment of each source-language word to at most one target-language word. The standard method to overcome this problem to use the model in both directions (interchanging the source and target languages) and applying heuristic-based combination techniques to produce a *refined alignment* (Och and Ney, 2000; Koehn et al., 2003)—henceforth referred to as "RA."

Several researchers have proposed algorithms for improving word alignment systems by injecting additional knowledge or combining different alignment models. These approaches include an enhanced HMM alignment model that uses part-of-speech tags (Toutanova et al., 2002), a log-linear combination of IBM translation models and HMM models (Och and Ney, 2003), techniques that rely on dependency relations (Cherry and Lin, 2003), and a log-linear combination of IBM Model 3 alignment probabilities, POS tags, and bilingual dictionary coverage (Liu et al., 2005). A common theme for these methods is the use of additional features for enriching the alignment process. These methods perform better than the IBM models and their variants but still tend to make similar errors because of the bias in their alignment modeling.

We adopt an approach that post-processes a given alignment using linguistically-oriented rules. The idea is similar to that of Ayan et al. (2004), where manually-crafted rules are used to correct alignment links related to language divergences. Our approach differs, however, in that the rules are extracted automatically—not manually—by examining an initial alignment and categorizing the errors according to features of the words.

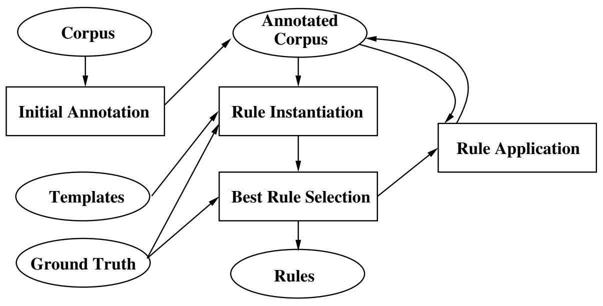


Figure 1: TBL Architecture

3 Transformation-based Learning

As shown in Figure 1, the input to TBL is an unannotated corpus that is first passed to an initial annotator and then iteratively updated through comparison to a manually-annotated reference set (or *ground truth*). On each iteration, the output of the previous iteration is compared against the ground truth, and an ordered list of transformation rules is learned that make the previous annotated data better resemble the ground truth.

A set of *rule templates* determines the space of allowable transformation rules. A rule template has two components: a triggering environment (condition of the rule) and a rewrite rule (action taken). On each iteration, these templates are instantiated with features of the constituents of the templates when the condition of the rule is satisfied.

This process eventually identifies all possible instantiated forms of the templates. Among all these possible rules, the transformation whose application results in the best score—according to some objective function—is identified. This transformation is added to the ordered list of transformation rules. The learning stops when there is no transformation that improves the current state of the data or a prespecified threshold is reached.

When presented with new data, the transformation rules are applied in the order that they were added to the list of transformations. The output of the system is the annotated data after all transformations are applied to the initial annotation.

4 Alignment Link Projection (ALP)

ALP is a TBL implementation that projects alignment links from an initial input alignment. We induce several variations of ALP by setting four parameters in different ways:

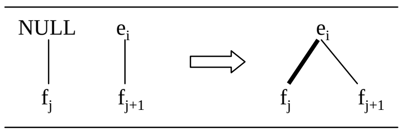


Figure 2: Graphical Representation of a Template

1. Initial alignment
2. Set of templates
3. Simple or generalized instantiation
4. Best rule selection

We describe each of these below using the following definitions and notation:

- $E = e_1, \ldots, e_i, \ldots, e_t$ is a sentence in language L_1 and $F = f_1, \ldots, f_j, \ldots, f_s$ is a sentence in language L_2.
- An *alignment link* (i, j) corresponds to a translational equivalence between e_i and f_j.
- A *neighborhood* of an alignment link (i, j)—denoted by $N(i, j)$—consists of 8 possible alignment links in a 3×3 window with (i, j) in the center of the window. Each element of $N(i, j)$ is called a *neighboring link* of (i, j).
- $nullE_A(i)$ is *true* if and only if e_i is not aligned to any word in F in a given alignment A. Similarly, $nullF_A(j)$ is *true* if and only if f_j is not aligned to any word in E in a given alignment A.

4.1 Initial Alignment

Any existing word-alignment system may be used for the initial annotation step of the TBL algorithm. For our experiments, we chose GIZA++ (Och and Ney, 2000) and the RA approach (Koehn et al., 2003)— the best known alignment combination technique— as our initial aligners.[1]

4.2 TBL Templates

Our templates consider consecutive words (of size 1, 2 or 3) in both languages. The condition portion of a TBL rule template tests for the existence of an alignment link between two words. The action portion involves the addition or deletion of an alignment link. For example, the rule template in Figure 2 is applicable only when a word (e_i) in one language is aligned to the second word (f_{j+1}) of a phrase (f_j, f_{j+1}) in the other language, and the first word (f_j) of the phrase is unaligned in the initial alignment. The action taken by this rule template is to add a link between e_i and f_j.[2]

ALP employs 3 different sets of templates to project new alignment links or delete existing links in a given alignment:

1. Expansion of the initial alignment according to another alignment
2. Deletion of spurious alignment links
3. Correction of multi-word (one-to-many or many-to-one) correspondences

Each of these is described below.

4.2.1 Expansion Templates

Expansion templates are used to extend an initial alignment given another alignment as the validation set. This approach is similar to the one used in the RA method in that it adds links based on knowledge about neighboring links, but it differs in that it *also* uses features of the words themselves to decide which neighboring links to add.

Our expansion templates are presented in Table 1. The first 8 templates add a new link to the initial alignment A if there is a neighboring link in the validation alignment V. The final two templates enforce the presence of at least two neighboring links in the validation set V before adding a new link.

Condition	**Action**
$(i,j) \in A, (i-1,j-1) \in V$	add $(i-1,j-1)$
$(i,j) \in A, (i-1,j) \in V$	add $(i-1,j)$
$(i,j) \in A, (i-1,j+1) \in V$	add $(i-1,j+1)$
$(i,j) \in A, (i,j-1) \in V$	add $(i,j-1)$
$(i,j) \in A, (i,j+1) \in V$	add $(i,j+1)$
$(i,j) \in A, (i+1,j-1) \in V$	add $(i+1,j-1)$
$(i,j) \in A, (i+1,j) \in V$	add $(i+1,j)$
$(i,j) \in A, (i+1,j+1) \in V$	add $(i+1,j+1)$
$(i-1,j-1) \in A, (i+1,j+1) \in A, (i,j) \in V$	add (i,j)
$(i+1,j-1) \in A, (i-1,j+1) \in A, (i,j) \in V$	add (i,j)

Table 1: Templates for Expanding the Alignment A According to a Validation Alignment V

4.2.2 Deletion Templates

Existing alignment algorithms (e.g., GIZA++) are biased toward aligning some words, especially infrequent ones, in one language to many words in the other language in order to minimize the number of unaligned words, even if many incorrect alignment

[1]We treat these initial aligners as black boxes.

[2]A thick line indicates an added link.

links are induced.[3] Deletion templates are useful for eliminating the resulting spurious links.

The basic idea is to remove alignment links that do not have a neighboring link if the word in question has already been aligned to another word. Table 2 lists two simple templates to clean up spurious links. We define the predicate $neighbor\_exists_A(i, j)$ to denote whether there is an alignment link in the neighborhood of the link (i, j) in a given alignment A. For example, the first template deletes spurious links for a particular word e_i in E.

Condition	**Action**
$(i, j) \in A, (i, k) \in A,$ $neighbor\_exists_A(i, j),$ $not(neighbor\_exists_A(i, k))$	del (i, k)
$(i, j) \in A, (k, j) \in A,$ $neighbor\_exists_A(i, j),$ $not(neighbor\_exists_A(k, j))$	del (e, j)

Table 2: Templates for Deleting Spurious Links in a Given Alignment A

4.2.3 Multi-Word Correction Templates

Current alignment algorithms produce one-to-one word correspondences quite successfully. However, accurate alignment of phrasal constructions (many-to-many correspondences) is still problematic. On the one hand, the ability to provide *fully* correct phrasal alignments is impaired by the occurrence of high-frequency function words and/or words that are not exact translations of the words in the other language. On the other hand, we have observed that most alignment systems are capable of providing *partially* correct phrasal alignments.[4]

Our templates for handling multi-word correspondences are grounded in the outcome of this finding. That is, we make the (frequently correct) assumption that at least one alignment link in a many-to-many correspondence is correctly identified in the initial alignment. Table 3 lists the templates for correcting alignment links in multi-word correspondences. The first five templates handle ($e_i \rightarrow f_j f_{j+1}$) correspondences, the next five handle ($e_i e_{i+1} \rightarrow f_j$) correspondences, the next four handle ($e_i e_{i+1} \rightarrow f_j f_{j+1}$) correspondences, and the final two handle ($e_{i-1} e_i e_{i+1} \rightarrow f_j$) and ($e_i \rightarrow f_{j-1} f_j f_{j+1}$) correspondences.

Condition	**Action**
$nullF_A(j), (i, j+1) \in A$	add (i, j)
$nullF_A(j+1), (i, j) \in A$	add $(i, j+1)$
$(i, j) \in A, (i, j+1) \in A$	del (i, j)
$(i, j) \in A, (i, j+1) \in A$	del $(i, j+1)$
$nullF_A(j), nullF_A(j+1)$	add (i, j), add $(i, j+1)$
$nullE_A(i), (i+1, j) \in A$	add (i, j)
$nullE_A(i+1), (i, j) \in A$	add $(i+1, j)$
$(i, j) \in A, (i+1, j) \in A$	del (i, j)
$(i, j) \in A, (i+1, j) \in A$	del $(i+1, j)$
$nullE_A(i), nullE_A(i+1)$	add (i, j) add $(i+1, j)$
$(i+1, j+1) \in A$ $nullE_A(i), nullF_A(j),$	add (i, j)
$(i, j) \in A, nullE_A(i+1),$ $nullF_A(j+1)$	add $(i+1, j+1)$
$(i, j) \in A, (i+1, j) \in A,$ $(i+1, j+1) \in A$	add $(i, j+1)$
$(i, j) \in A, (i, j+1) \in A,$ $(i+1, j+1) \in A$	add $(i+1, j)$
$(i-1, j) \in A, (i+1, j) \in A$ $nullE_A(i)$	add (i, j)
$(i, j-1) \in A, (i, j+1) \in A$ $nullF_A(j)$	add (i, j)

Table 3: Templates for Handling Multi-Word Correspondences in a Given Alignment A

Condition	**Action**
$(i, j) \in A$	del (i, j)
$nullE_A(i), nullF_A(j)$	add (i, j)

Table 4: Templates for Correcting One-to-One Correspondences in a Given Alignment A

The alignment rules given above may introduce errors that require additional cleanup. Thus, we introduce two simple templates (shown in Table 4) to accommodate the deletion or addition of links between a single pair of words.

4.3 Instantiation of Templates

ALP starts with a set of templates and an initial alignment and attempts to instantiate the templates during the learning process. The templates can be instantiated using two methods: Simple (a word is instantiated with a specific feature) or Generalized (a word is instantiated using a special keyword `any-`

[3]This is a well-known characteristic of statistical alignment systems—motivated by the need to ensure a target-word translation e_i for each source word f_j while modeling $p(F|E)$ —for downstream MT.

[4]Specifically, we conducted a preliminary study using 40 manually-aligned English-Spanish sentences from a mixed corpus (UN + Bible + FBIS) as our gold standard. We found that, in most cases where the human annotator aligned one word to two words, an existing alignment system identified at least one of the two alignment links correctly.

`thing`).

ALP requires only a small amount of manually aligned data for this process—a major strength of the system. However, if we were to instantiate the templates with the actual words of the manual alignment, the frequency counts (from such a small data set) would not be high enough to derive reasonable generalizations. Thus, ALP adds new links based on linguistic features of words, rather than the words themselves. Using these features is what sets ALP apart from systems like the RA approach. Specifically, three features are used to instantiate the templates:

- **POS tags on both sides**: We assign POS tags using the MXPOST tagger (Ratnaparkhi, 1996) for English and Chinese, and Connexor for Spanish.
- **Dependency relations**: ALP utilizes dependencies for a better generalization—if a dependency parser is available in either language. In our experiments, we used a dependency parser only in English (a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies) but not in the other language.
- **A set of closed-class words**: We use 16 different classes, 9 of which are different semantic verb classes while the other 7 are function words, prepositions, and complementizers.[5]

If both POS tags and dependency relations are available, they can be used together to instantiate the templates. That is, a word can be instantiated in a TBL template with: (1) a POS tag (e.g., Noun, Adj); (2) a relation (e.g., Subj, Obj); (3) a parameter class (e.g., Change of State); or (4) different subsets of (1)–(3). We also employ a more generalized form of instantiation, where words in the templates may match the keyword `anything`.

4.4 Best Rule Selection

The rules are selected using two different metrics: The accuracy of the rule or the overall impact of the application of the rule on the entire data.

Two different mechanisms may be used for selecting the best rule after generating all possible instantiations of templates:

[5]These are based on the parameter classes of (Dorr et al., 2002).

1. **Rule Accuracy:** The goal is to minimize the errors introduced by the application of a transformation rule. To measure accuracy of a rule r, we use $good(r) - 2 \times bad(r)$, where $good(r)$ is the number of alignment links that are corrected by the rule, and $bad(r)$ is the number of incorrect alignment links produced.
2. **Overall impact on the training data:** The accuracy mechanism (above) is useful for biasing the system toward higher precision. However, if the overall system is evaluated using a metric other than precision (e.g., recall), the accuracy mechanism may not guarantee that the best rule is chosen at each step. Thus, we choose the best rule according to the evaluation metric to be used for the overall system.

5 Experiments and Results

This section describes our evaluation of ALP variants using different combinations of settings of the four parameters described above. The two language pairs examined are English-Spanish and English-Chinese.

5.1 Evaluation Metrics

Let A be the set of alignment links for a set of sentences. We take S to be the set of sure alignment links and P be the set of probable alignment links (in the gold standard) for the same set of sentences. Precision (Pr), recall (Rc) and alignment error rate (AER) are defined as follows:

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

A manually aligned corpus is used as our gold standard. For English-Spanish data, the manual annotation was done by a bilingual English-Spanish speaker. Every link in the English-Spanish gold standard is considered a sure alignment link.

For English-Chinese, we used 2002 NIST MT evaluation test set, and each sentence pair was aligned by two native Chinese speakers who are fluent in English. Each alignment link appearing in both annotations was considered a sure link, and links appearing in only one set were judged as probable. The annotators were not aware of the specifics of our approach.

5.2 Evaluation Data

We evaluated ALP using 5-fold cross validation on two different data sets:

1. A set of 199 English-Spanish sentence pairs (nearly 5K words on each side) from a mixed corpus (UN + Bible + FBIS).
2. A set of 491 English-Chinese sentence pairs (nearly 13K words on each side) from 2002 NIST MT evaluation test set.

We divided the pairs of sentences randomly into 5 groups. Then, for each fold, we used 4 groups as the ground truth (for training), and used the other group as our gold standard (for evaluation). This process was repeated 5 times so that each sentence pair was tested exactly once. We computed precision, recall and error rate on the entire set for each data set.[6]

For an initial alignment, we used GIZA++ in both directions (E-to-F and F-to-E, where F is either Chinese (C) or Spanish (S)), and also two different combined alignments: intersection of E-to-F and F-to-E; and RA using a heuristic combination approach called *grow-diag-final* (Koehn et al., 2003).

For the English-Spanish experiments, GIZA++ was trained on 48K sentence pairs from a mixed corpus (UN + Bible + FBIS), with nearly 1.2M of words on each side, using 10 iterations of Model 1, 5 iterations of HMM and 5 iterations of Model 4. For the English-Chinese experiments, we used 107K sentence pairs from FBIS corpus (nearly 4.1M English and 3.3M Chinese words) to train GIZA++, using 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

5.3 Results for English-Spanish

For our initial alignments we used: (1) Intersection of GIZA++ English-to-Spanish and Spanish-to-English; (2) GIZA++ English-to-Spanish; (3) GIZA++ Spanish-to-English; and (4) RA. Of these, RA is the best, with an error rate of 21.2%. For ease of comparison, the RA score appears in all result tables below.

[6]The number of alignment links varies over each fold. Therefore, we chose to evaluate all data at once instead of evaluating on each fold and then averaging.

Tables 5–7 compare ALP to each of these four alignments using different settings of 4 parameters: ALP[*IA, T, I, BRS*], where *IA* is the initial alignment, T is the set of templates, I is the instantiation method, and *BRS* is the metric for the best rule selection at each iteration. T_E is the set of expansion templates from Table 1, T_D is the set of deletion templates from Table 2, and T_{MW} is the set of multi-word templates from Table 3 (supplemented with templates from Table 4).

As mentioned in Section 4.3, we use two instantiation methods: (1) simple instantiation (sim), where the words are instantiated using a specific POS tag, relation, parameter class or combination of those; and (2) generalized instantiation (gen), where the words can be instantiated using the keyword `anything`. Two different metrics are used to select the best rule: The accuracy of the rule (acc) and the AER on the entire training data after applying the rule (aer).[7]

We performed statistical significance tests using two-tailed paired t-tests. Unless otherwise indicated, the differences between ALP and initial alignments (for all ALP variations and all initial alignments) were found to be statistically significant within the 95% confidence interval. Moreover, the differences among ALP variations themselves were statistically significant within 95% confidence interval.

Using Intersection as Initial Alignment We ran ALP using the intersection of GIZA++ (E-to-S) and GIZA++(S-to-E) alignments as the initial alignment in two different ways: (1) With T_E using the union of the unidirectional GIZA++ alignments as the validation set, and (2) with T_D and T_{MW} applied one after another. Table 5 presents the precision, recall and AER results.

Alignments	**Pr**	**Rc**	**AER**
Intersection (Int)	**98.2**	59.6	25.9
ALP[Int, T_E, gen, aer]	90.9	69.9	21.0
ALP[$Int, (T_D, T_{MW}), gen, aer$]	88.8	72.3	**20.3**
RA	83.8	**74.4**	21.2

Table 5: ALP Results Using GIZA++ Intersection as Initial Alignment for English-Spanish

Using the expansion templates (T_E) against a val-

[7]We use only sure alignment links as the ground truth to learn rules inside ALP. Therefore, AER here refers to the AER of sure alignment links.

Alignments	**Pr**	**Rc**	**AER**
E-to-S	**87.0**	67.0	24.3
ALP[E-to-S,$(T_D, T_{MW}), gen, aer$]	85.6	**76.4**	**19.3**
S-to-E	**88.0**	67.5	23.6
ALP[S-to-E,$(T_D, T_{MW}), gen, aer$]	87.1	**76.7**	**18.4**
RA	83.8	74.4	21.2

Table 6: ALP Results Using GIZA++ (Each Direction) as Initial Alignment for English-Spanish

idation set produced results comparable to the RA method. The major difference is that ALP resulted in a much higher precision but in a lower recall because ALP is more selective in adding a new link during the expansion stage. This difference is due to the additional constraints provided by word features. The version of ALP that applies deletion (T_D) and multi-word (T_{MW}) templates sequentially achieves lower recall but higher precision than RA. In the best case, ALP achieves a statistically significant relative reduction of 21.6% in AER over the Intersection alignment. When compared to RA, ALP achieves a lower AER but the difference is not significant.

Using Unidirectional GIZA++ Alignments as Initial Alignment In a second set of experiments, we applied ALP to the unidirectional GIZA++ alignments, using deletion (T_D) and multi-word (T_{MW}) templates, generalized instantiation, and AER for the best rule selection. Table 6 presents the precision, recall and AER results.

For both directions, ALP achieves a lower precision but much higher recall than that of the initial unidirectional alignment. Overall, there was a relative reduction of 20.6–22.0% in AER. When compared to RA, the version of ALP that uses unidirectional GIZA++ alignments brings about significant reductions in AER: 9.0% relative reduction in one direction and 13.2% relative reduction in the other direction.

Using RA as Initial Alignment In a third experiment, we compared RA with variations of ALP using RA as the initial alignment. We used the templates in two different ways: (1) with a combination of T_D and T_{MW} (i.e., $T_D \cup T_{MW}$), and (2) with two consecutive runs of ALP, first with T_D and then with T_{MW} using the output of the first run as the initial annotation in the second run (i.e., T_D, T_{MW}). Table 7 presents precision, recall and AER results, using different methods for template instantiation and

Alignments	**Pr**	**Rc**	**AER**
ALP[$RA, (T_D, T_{MW}), sim, acc$]	87.8	77.7	17.6
ALP[$RA, (T_D, T_{MW}), sim, aer$]	**87.9**	79.0	16.8
ALP[$RA, (T_D \cup T_{MW}), gen, aer$]	86.2	80.0	17.0
ALP[$RA, (T_D, T_{MW}), gen, aer$]	86.9	**80.5**	**16.4**
RA	83.8	74.4	21.2

Table 7: ALP Results Using RA as Initial Alignment for English-Spanish

best rule selection.

The results indicate that using AER is better than using accuracy for choosing the best rule. Using generalized instantiation instead of simple instantiation results in a better AER. Running ALP with deletion (T_D) templates followed by multi-word (T_{MW}) templates results in a lower AER than running ALP only once with combined templates.

The highest performing variant of ALP, shown in the fourth line of the table, uses RA as the initial alignment, template sets T_D, T_{MW}, generalized instantiation, and AER for best rule selection. This variant is significantly better than RA, with a 22.6% relative reduction in AER. When compared to the unidirectional alignments (E-to-S and S-to-E) given in Table 6, this variant of ALP yields nearly the same precision (around 87.0%) but a 19.2% relative improvement in recall. The overall relative reduction in AER is 30.5% in the S-to-E direction and 32.5% in the E-to-S direction.

5.4 Results for English-Chinese

Our experiments for English-Chinese were designed with a similar structure to that of English-Spanish, i.e., the same four initial alignments. Once again, RA performs the best out of these initial alignments, with an error rate of 29.7%. The results of the initial alignments, and variations of ALP based on different initial alignments are shown in Table 8. For brevity, we include only the ALP parameter settings resulting in the best configurations from the English-Spanish experiments. For learning rules from the templates, we used only the sure alignment links as the ground truth while learning rules inside ALP.

On the English-Chinese data, ALP yields significantly lower error rates with respect to the initial alignments. When ALP is run with the intersection of two GIZA++ alignments, the relative reduction is 5.4% in AER. When ALP is run with E-to-C as initial alignment, the relative reduction in AER is 13.4%. For the other direction, ALP produces a rel-

Alignments	Pr	Rc	AER
Intersection (Int)	**94.8**	53.6	31.2
ALP[$Int, (T_D, T_{MW}), gen, aer$]	91.7	**56.8**	**29.5**
E-to-C	70.4	**68.3**	30.7
ALP[E-to-C,$(T_D, T_{MW}), gen, aer$]	**79.1**	68.1	**26.6**
C-to-E	66.0	**69.8**	32.2
ALP[C-to-E,$(T_D, T_{MW}), gen, aer$]	**83.3**	66.0	**26.2**
RA	61.9	**82.6**	29.7
ALP[RA,$(T_D, T_{MW}), gen, aer$]	**82.1**	72.7	**22.8**

Table 8: ALP Results Using Different Initial Alignments for English-Chinese

ative reduction of 18.6% in AER. Finally, when RA is given to ALP as an initial alignment, ALP results in a relative reduction of 23.2% in AER. When compared to RA, all variations of ALP, except the one starting with the intersection, yield statistically significantly lower AER. Another important finding is that ALP yields significantly higher precision than the initial alignments but usually lower recall.

6 Conclusion

We have presented ALP, a new approach that refines alignments by identifying the types of errors made by existing alignment systems and correcting them. Our approach adapts TBL to the problem of word-level alignment by examining word features as well as neighboring links. We use POS tags, closed-class words in both languages, and dependency relations in one language to classify the errors made by the initial alignment system. We show that ALP yields at least a 22.6% relative reduction on English-Spanish data and 23.2% relative reduction on English-Chinese data in alignment error rate over that of the best performing system.

We should note that ALP is not a stand-alone word alignment system but a supervised learning approach to improve already existing alignment systems. ALP takes advantage of clustering of alignment links to project new links given a reasonable initial alignment. We have shown that ALP is quite successful in projecting alignment links for two different languages—Spanish and Chinese.

Statistical alignment systems are more successful with increasing amount of training data. Whether ALP improves the statistical alignment systems when they are trained on more data is an interesting research problem, which we plan to tackle in future.

Finally, we will evaluate the improved alignments in the context of an end-to-end application, such as machine translation.

Acknowledgments This work has been supported, in part, by ONR MURI Contract FCPO.810548265, Cooperative Agreement DAAD190320020, and NSF ITR Grant IIS-0326553.

References

Necip F. Ayan, Bonnie J. Dorr, and Nizar Habash. 2004. Multi-Align: Combining linguistic and statistical techniques to improve alignments for adaptable MT. In *Proceedings of AMTA'2004*, pages 17–26.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Eric Brill. 1996. Learning to parse with transformations. In *Recent Advances in Parsing Technology*. Kluwer Academic Publishers.

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.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL'2003*, pages 88–95.

Micheal Collins. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of ACL'1997*.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of ACL'2002*.

Bonnie J. Dorr, Lisa Pearl, Rebecca Hwa, and Nizar Habash. 2002. DUSTer: A method for unraveling cross-language divergences for statistical word–level alignment. In *Proceedings of AMTA'2002*.

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

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL'2005*.

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

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL'2000*, pages 440–447.

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

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP'1996*.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of EMNLP'2002*, pages 87–94.

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

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT'2001*, pages 109–116.

Predicting Sentences using N-Gram Language Models

Steffen Bickel, Peter Haider, and Tobias Scheffer
Humboldt-Universität zu Berlin
Department of Computer Science
Unter den Linden 6, 10099 Berlin, Germany
{bickel, haider, scheffer}@informatik.hu-berlin.de

Abstract

We explore the benefit that users in several application areas can experience from a "tab-complete" editing assistance function. We develop an evaluation metric and adapt N-gram language models to the problem of predicting the subsequent words, given an initial text fragment. Using an instance-based method as baseline, we empirically study the predictability of call-center emails, personal emails, weather reports, and cooking recipes.

1 Introduction

Prediction of user behavior is a basis for the construction of assistance systems; it has therefore been investigated in diverse application areas. Previous studies have shed light on the predictability of the next unix command that a user will enter (Motoda and Yoshida, 1997; Davison and Hirsch, 1998), the next keystrokes on a small input device such as a PDA (Darragh and Witten, 1992), and of the translation that a human translator will choose for a given foreign sentence (Nepveu et al., 2004).

We address the problem of predicting the subsequent words, given an initial fragment of text. This problem is motivated by the perspective of assistance systems for repetitive tasks such as answering emails in call centers or letters in an administrative environment. Both instance-based learning and N-gram models can conjecture completions of sentences. The use of N-gram models requires the application of the Viterbi principle to this particular decoding problem.

Quantifying the benefit of editing assistance to a user is challenging because it depends not only on an observed distribution over documents, but also on the reading and writing speed, personal preference, and training status of the user. We develop an evaluation metric and protocol that is practical, intuitive, and independent of the user-specific trade-off between keystroke savings and time lost due to distractions. We experiment on corpora of service-center emails, personal emails of an Enron executive, weather reports, and cooking recipes.

The rest of this paper is organized as follows. We review related work in Section 2. In Section 3, we discuss the problem setting and derive appropriate performance metrics. We develop the N-gram-based completion method in Section 4. In Section 5, we discuss empirical results. Section 6 concludes.

2 Related Work

Shannon (1951) analyzed the predictability of sequences of letters. He found that written English has a high degree of redundancy. Based on this finding, it is natural to ask whether users can be supported in the process of writing text by systems that predict the intended next keystrokes, words, or sentences. Darragh and Witten (1992) have developed an *interactive keyboard* that uses the sequence of past keystrokes to predict the most likely succeeding keystrokes. Clearly, in an unconstrained application context, keystrokes can only be predicted with limited accuracy. In the specific context of entering URLs, completion predictions are commonly provided by web browsers (Debevc et al., 1997).

Motoda and Yoshida (1997) and Davison and Hirsch (1998) developed a Unix shell which predicts the command stubs that a user is most likely to enter, given the current history of entered commands. Korvemaker and Greiner (2000) have developed this idea into a system which predicts entire command lines. The Unix command prediction problem has also been addressed by Jacobs and Blockeel (2001) who infer macros from frequent command sequences and predict the next command using variable memory Markov models (Jacobs and Blockeel, 2003).

In the context of *natural language*, several typing assistance tools for apraxic (Garay-Vitoria and Abascal, 2004; Zagler and Beck, 2002) and dyslexic (Magnuson and Hunnicutt, 2002) persons have been developed. These tools provide the user with a list of possible word completions to select from. For these users, scanning and selecting from lists of proposed words is usually more efficient than typing. By contrast, scanning and selecting from many displayed options can slow down skilled writers (Langlais et al., 2002; Magnuson and Hunnicutt, 2002).

Assistance tools have furthermore been developed for translators. Computer aided translation systems combine a translation and a language model in order to provide a (human) translator with a list of suggestions (Langlais et al., 2000; Langlais et al., 2004; Nepveu et al., 2004). Foster et al. (2002) introduce a model that adapts to a user's typing speed in order to achieve a better trade-off between distractions and keystroke savings. Grabski and Scheffer (2004) have previously developed an indexing method that efficiently retrieves the sentence from a collection that is most similar to a given initial fragment.

3 Problem Setting and Evaluation

Given an initial text fragment, a predictor that solves the sentence completion problem has to conjecture *as much of the sentence that the user currently intends to write,* as is possible with high confidence—preferably, but not necessarily, the entire remainder.

The perceived benefit of an assistance system is highly subjective, because it depends on the expenditure of time for scanning and deciding on suggestions, and on the time saved due to helpful assistance. The user-specific benefit is influenced by quantitative factors that we can measure. We construct a system of two conflicting performance indicators: our definition of *precision* quantifies the inverse risk of unnecessary distractions, our definition of *recall* quantifies the rate of keystroke savings.

For a given sentence fragment, a completion method may – but need not – cast a completion conjecture. Whether the method suggests a completion, and how many words are suggested, will typically be controlled by a confidence threshold. We consider the entire conjecture to be falsely positive if at least one word is wrong. This harsh view reflects previous results which indicate that selecting, and then editing, a suggested sentence often takes longer than writing that sentence from scratch (Langlais et al., 2000). In a conjecture that is entirely accepted by the user, the entire string is a true positive. A conjecture may contain only a part of the remaining sentence and therefore the *recall*, which refers to the length of the missing part of the current sentence, may be smaller than 1.

For a given test collection, precision and recall are defined in Equations 1 and 2. *Recall* equals the fraction of saved keystrokes (disregarding the interface-dependent single keystroke that is most likely required to accept a suggestion); *precision* is the ratio of characters that the users have to scan for each character they accept. Varying the confidence threshold of a sentence completion method results in a *precision recall curve* that characterizes the system-specific trade-off between *keystroke savings* and *unnecessary distractions*.

$$Precision = \frac{\sum_{\text{accepted completions}} \text{string length}}{\sum_{\text{suggested completions}} \text{string length}} \quad (1)$$

$$Recall = \frac{\sum_{\text{accepted completions}} \text{string length}}{\sum_{\text{all queries}} \text{length of missing part}} \quad (2)$$

4 Algorithms for Sentence Completion

In this section, we derive our solution to the sentence completion problem based on linear interpolation of N-gram models. We derive a k best Viterbi decoding algorithm with a confidence-based stopping criterion which conjectures the words that most likely succeed an initial fragment. Additionally, we briefly discuss an instance-based method that provides an alternative approach and baseline for our experiments.

In order to solve the sentence completion problem with an N-gram model, we need to find the most likely word sequence $w_{t+1}, \ldots, w_{t+T}$ given a word N-gram model and an initial sequence $w_1, \ldots, w_t$ (Equation 3). Equation 4 factorizes the joint probability of the missing words; the N-th order Markov assumption that underlies the N-gram model simplifies this expression in Equation 5.

$$\begin{aligned} & \operatorname*{argmax}_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T} | w_1, \ldots, w_t) && (3) \\ = & \operatorname*{argmax}_{w_{t+1},\ldots,w_{t+T}} \prod_{j=1}^{T} P(w_{t+j} | w_1, \ldots, w_{t+j-1}) && (4) \\ = & \operatorname*{argmax} \prod_{j=1}^{T} P(w_{t+j} | w_{t+j-N+1}, \ldots, w_{t+j-1}) && (5) \end{aligned}$$

The individual factors of Equation 5 are provided by the model. The Markov order N has to balance sufficient context information and sparsity of the training data. A standard solution is to use a weighted linear mixture of N-gram models, $1 \leq n \leq N$, (Brown et al., 1992). We use an EM algorithm to select mixing weights that maximize the generation probability of a tuning set of sentences that have not been used for training.

We are left with the following questions: (a) how can we decode the most likely completion *efficiently*; and (b) how many words should we predict?

4.1 Efficient Prediction

We have to address the problem of finding the most likely completion, $\operatorname{argmax}_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T} | w_1, \ldots, w_t)$ *efficiently*, even though the size of the *search space* grows exponentially in the number of predicted words.

We will now identify the recursive structure in Equation 3; this will lead us to a Viterbi algorithm that retrieves the most likely word sequence. We first define an auxiliary variable $\delta_{t,s}(w'_1, \ldots, w'_N | w_{t-N+2}, \ldots, w_t)$ in Equation 6; it quantifies the greatest possible probability over all arbitrary word sequences $w_{t+1}, \ldots, w_{t+s}$, followed by the word sequence $w_{t+s+1} = w'_1, \ldots, w_{t+s+N} = w'_N$, conditioned on the initial word sequence $w_{t-N+2}, \ldots, w_t$.

In Equation 7, we factorize the last transition and utilize the N-th order Markov assumption. In Equation 8, we split the maximization and introduce a new random variable w'_0 for w_{t+s}. We can now refer to the definition of δ and see the recursion in Equation 9: $\delta_{t,s}$ depends only on $\delta_{t,s-1}$ and the N-gram model probability $P(w'_N | w'_1, \ldots, w'_{N-1})$.

$$\begin{aligned} & \delta_{t,s}(w'_1, \ldots, w'_N | w_{t-N+2}, \ldots, w_t) && (6) \\ & = \max_{w_{t+1},\ldots,w_{t+s}} P(w_{t+1}, \ldots, w_{t+s}, w_{t+s+1} = w'_1, \\ & \qquad \ldots, w_{t+s+N} = w'_N | w_{t-N+2}, \ldots, w_t) \\ & = \max_{w_{t+1},\ldots,w_{t+s}} P(w'_N | w'_1, \ldots, w'_{N-1}) && (7) \\ & \qquad P(w_{t+1}, \ldots, w_{t+s}, w_{t+s+1} = w'_1, \\ & \qquad \ldots, w_{t+s+N-1} = w'_{N-1} | w_{t-N+2}, \ldots, w_t) \\ & = \max_{w'_0} \max_{w_{t+1},\ldots,w_{t+s-1}} P(w'_N | w'_1, \ldots, w'_{N-1}) && (8) \\ & \qquad P(w_{t+1}, \ldots, w_{t+s-1}, w_{t+s} = w'_0, \\ & \qquad \ldots, w_{t+s+N-1} = w'_{N-1} | w_{t-N+2}, \ldots, w_t) \\ & = \max_{w'_0} P(w'_N | w'_1, \ldots, w'_{N-1}) && (9) \\ & \qquad \delta_{t,s-1}(w'_0, \ldots, w'_{N-1} | w_{t+N-2}, \ldots, w_t) \end{aligned}$$

Exploiting the N-th order Markov assumption, we can now express our target probability (Equation 3) in terms of δ in Equation 10.

$$\begin{aligned} & \max_{w_{t+1},\ldots,w_{t+T}} P(w_{t+1}, \ldots, w_{t+T} | w_{t-N+2}, \ldots, w_t) && (10) \\ & = \max_{w'_1,\ldots,w'_N} \delta_{t,T-N}(w'_1, \ldots, w'_N | w_{t-N+2}, \ldots, w_t) \end{aligned}$$

The last N words in the most likely sequence are simply the $\operatorname{argmax}_{w'_1,\ldots,w'_N} \delta_{t,T-N}(w'_1, \ldots, w'_N | w_{t-N+2}, \ldots, w_t)$. In order to collect the preceding most likely words, we define an auxiliary variable Ψ in Equation 11 that can be determined in Equation 12. We have now found a Viterbi algorithm that is linear in T, the completion length.

$$\begin{aligned} & \Psi_{t,s}(w'_1, \ldots, w'_N | w_{t-N+2}, \ldots, w_t) && (11) \\ & = \operatorname*{argmax}_{w_{t+s}} \max_{w_{t+1},\ldots,w_{t+s-1}} P(w_{t+1}, ..., w_{t+s}, w_{t+s+1} = w'_1, ..., \\ & \qquad w_{t+s+N} = w'_N | w_{t-N+2}, ..., w_t) \\ & = \operatorname*{argmax}_{w'_0} \delta_{t,s-1}(w'_0, \ldots, w'_{N-1} | w_{t-N+2}, \ldots, w_t) && (12) \\ & \qquad P(w'_N | w'_1, \ldots, w'_{N-1}) \end{aligned}$$

The Viterbi algorithm starts with the most recently entered word w_t and moves iteratively into the future. When the N-th token in the highest scored δ is a period, then we can stop as our goal is only to predict (parts of) the current sentence. However, since there is no guarantee that a period will eventually become the most likely token, we use an absolute confidence threshold as additional criterion: when the highest δ score is below a threshold θ, we stop the Viterbi search and fix T.

In each step, Viterbi stores and updates $|\text{vocabulary size}|^N$ many δ values—unfeasibly many except for very small N. Therefore, in Table 1 we develop a Viterbi beam search algorithm which is linear in T and in the beam width. Beam search cannot be guaranteed to always find the most likely word sequence: When the globally most likely sequence $w^*_{t+1}, \ldots, w^*_{t+T}$ has an initial subsequence $w^*_{t+1}, \ldots, w^*_{t+s}$ which is not among the k most likely sequences of length s, then that optimal sequence is not found.

Table 1: Sentence completion with Viterbi beam search algorithm.

```
Input: $N$-gram language model, initial sentence fragment $w_1, \ldots, w_t$, beam width $k$, confidence threshold $\theta$.

1. Viterbi initialization:
   Let $\delta_{t,-N}(w_{t-N+1}, \ldots, w_t | w_{t-N+1}, \ldots, w_t) = 1$;
   let $s = -N + 1$;
   $beam(s-1) = \{\delta_{t,-N}(w_{t-N+1}, \ldots, w_t | w_{t-N+1}, \ldots, w_t)\}$.
2. Do Viterbi recursion until break:
   (a) For all $\delta_{t,s-1}(w'_0, \ldots, w'_{N-1} | \ldots)$ in $beam(s-1)$, for all $w_N$ in vocabulary, store $\delta_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 9) in $beam(s)$ and calculate $\Psi_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 12).
   (b) If $\text{argmax}_{w_N} \max_{w'_1, \ldots, w'_{N-1}} \delta_{t,s}(w'_1, \ldots, w'_N | \ldots) = period$ then break.
   (c) If $\max \delta_{t,s}(w'_1, \ldots, w'_N | w_{t-N+1}, \ldots, w_t) < \theta$ then decrement $s$; break.
   (d) Prune all but the best $k$ elements in $beam(s)$.
   (e) Increment $s$.
3. Let $T = s + N$. Collect words by path backtracking:
      $(w^*_{t+T-N+1}, \ldots, w^*_{t+T}) = \text{argmax}\, \delta_{t,T-N}(w'_1, \ldots, w'_N | \ldots)$.
   For $s = T - N \ldots 1$:
      $w^*_{t+s} = \Psi_{t,s}(w^*_{t+s+1}, \ldots, w^*_{t+s+N} | w_{t-N+1}, \ldots, w_t)$.

Return $w^*_{t+1}, \ldots, w^*_{t+T}$.
```

4.2 Instance-based Sentence Completion

An alternative approach to sentence completion based on N-gram models is to retrieve, from the training collection, the sentence that starts most similarly, and use its remainder as a completion hypothesis. The cosine similarity of the TFIDF representation of the initial fragment to be completed, and an equally long fragment of each sentence in the training collection gives both a selection criterion for the nearest neighbor and a confidence measure that can be compared against a threshold in order to achieve a desired precision recall balance.

A straightforward implementation of this nearest neighbor approach becomes infeasible when the training collection is large because too many training sentences have to be processed. Grabski and Scheffer (2004) have developed an indexing structure that retrieves the most similar (using cosine similarity) sentence fragment in sub-linear time. We use their implementation of the instance-based method in our experimentation.

5 Empirical Studies

we investigate the following questions. (a) How does sentence completion with N-gram models compare to the instance-based method, both in terms of precision/recall and computing time? (b) How well can N-gram models complete sentences from collections with diverse properties?

Table 2 gives an overview of the four document collections that we use for experimentation. The first collection has been provided by a large online store and contains emails sent by the service center in reply to customer requests (Grabski and Scheffer, 2004). The second collection is an excerpt of the recently disclosed email correspondence of Enron's management staff (Klimt and Yang, 2004). We use 3189 personal emails sent by Enron executive Jeff Dasovich; he is the individual who sent the largest number of messages within the recording period.

The third collection contains textual daily weather reports for five years from a weather report provider on the Internet. Each report comprises about 20 sentences. The last collection contains about 4000 cooking recipes; this corpus serves as an example of a set of thematically related documents that might be found on a personal computer.

We reserve 1000 sentences of each data set for testing. As described in Section 4, we split the remaining sentences in training (75%) and tuning

Table 2: Evaluation data collections.

Name	Language	#Sentences	Entropy
service center	German	7094	1.41
Enron emails	English	16363	7.17
weather reports	German	30053	4.67
cooking recipes	German	76377	4.14

(25%) sets. We mix N-gram models up to an order of five and estimate the interpolation weights (Section 4). The resulting weights are displayed in Figure 1. In Table 2, we also display the entropy of the collections based on the interpolated 5-gram model. This corresponds to the average number of bits that are needed to code each word given the preceding four words. This is a measure of the intrinsic redundancy of the collection and thus of the predictability.

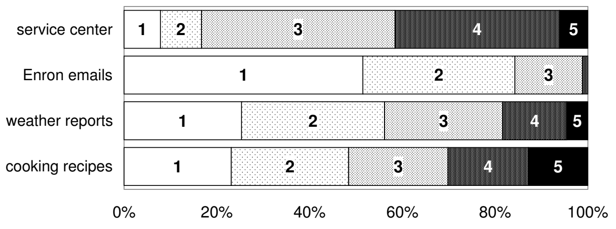


Figure 1: N-gram interpolation weights.

Our evaluation protocol is as follows. The beam width parameter k is set to 20. We randomly draw 1000 sentences and, within each sentence, a position at which we split it into initial fragment and remainder to be predicted. A human evaluator is presented both, the actual sentence from the collection and the initial fragment plus current completion conjecture. For each initial fragment, we first cast the most likely single word prediction and ask the human evaluator to judge whether they would accept this prediction (without any changes), given that they intend to write the actual sentence. We increase the length of the prediction string by one additional word and recur, until we reach a period or exceed the prediction length of 20 words.

For each judged prediction length, we record the confidence measure that would lead to that prediction. With this information we can determine the results for all possible threshold values of θ. To save evaluation time, we consider all predictions that are identical to the actual sentence as correct and skip those predictions in the manual evaluation.

We will now study how the N-gram method compares to the instance-based method. Figure 2 compares the precision recall curves of the two methods. Note that the maximum possible recall is typically much smaller than 1: recall is a measure of the keystroke savings, a value of 1 indicates that the user saves *all* keystrokes. Even for a confidence threshold of 0, a recall of 1 is usually not achievable.

Some of the precision recall curves have a concave shape. Decreasing the threshold value increases the number of predicted words, but it also increases the risk of at least one word being wrong. In this case, the entire sentence counts as an incorrect prediction, causing a decrease in both, precision and recall. Therefore – unlike in the standard information retrieval setting – recall does not increase monotonically when the threshold is reduced.

For three out of four data collections, the instance-based learning method achieves the highest maximum recall (whenever this method casts a conjecture, the entire remainder of the sentence is predicted—at a low precision), but for nearly all recall levels the N-gram model achieves a much higher precision. For practical applications, a high precision is needed in order to avoid distracting, wrong predictions. Varying the threshold, the N-gram model can be tuned to a wide range of different precision recall trade-offs (in three cases, precision can even reach 1), whereas the confidence threshold of the instance-based method has little influence on precision and recall.

We determine the standard error of the precision for the point of maximum F1-measure. For all data collections and both methods the standard error is below 0.016. Correct and incorrect prediction examples are provided in Table 3 for the service center data set, translated from German into English. The confidence threshold is adjusted to the value of maximum F1-measure. In two of these cases, the prediction nicely stops at fairly specific terms.

How do precision and recall depend on the string length of the initial fragment and the string length of the completion cast by the systems? Figure 3 shows the relationship between the length of the initial fragment and precision and recall. The performance of the instance-based method depends crucially on a long initial fragment. By contrast, when

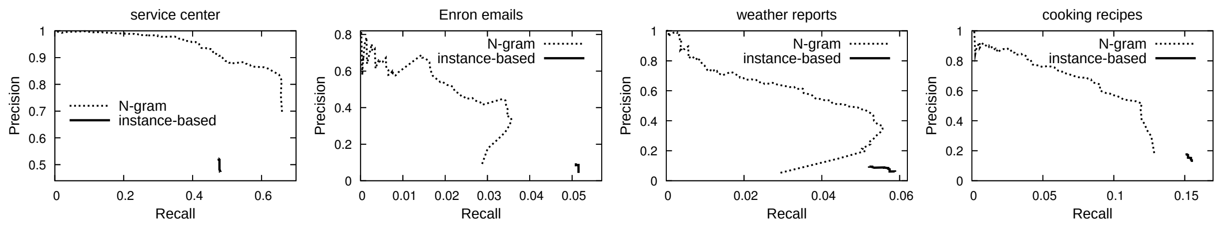


Figure 2: Precision recall curves for N-gram and instance-based methods of sentence completion.

Table 3: Prediction examples for service center data.

Initial fragment (bold face) and intended, missing part	Prediction
Please complete your address.	your address.
Kindly excuse the incomplete shipment.	excuse the
Our supplier notified us that the pants are undeliverable.	notified us that the
The mentioned order is not in our system.	not in our system.
We recommend that you write down your login name and password.	that you write down your login name and password.
The value will be accounted for in your invoice.	be accounted for in your invoice.
Please excuse the delay.	delay.
Please excuse our mistake.	the delay.
If this is not the case give us a short notice.	us your address and customer id.

the fragment length exceeds four with the N-gram model, then this length and the accuracy are nearly independent; the model considers no more than the last four words in the fragment.

Figure 4 details the relation between string length of the prediction and precision/recall. We see that we can reach a constantly high precision over the entire range of prediction lengths for the service center data with the N-gram model. For the other collections, the maximum prediction length is 3 or 5 words in comparison to much longer predictions cast by the nearest neighbor method. But in these cases, longer predictions result in lower precision.

How do instance-based learning and N-gram completion compare in terms of computation time? The Viterbi beam search decoder is linear in the prediction length. The index-based retrieval algorithm is constant in the prediction length (except for the final step of *displaying* the string which is linear but can be neglected). This is reflected in Figure 5 (left) which also shows that the absolute decoding time of both methods is on the order of few milliseconds on a PC. Figure 5 (right) shows how prediction time grows with the training set size.

We experiment on four text collections with diverse properties. The N-gram model performs remarkably on the service center email collection. Users can save 60% of their keystrokes with 85% of all suggestions being accepted by the users, or save 40% keystrokes at a precision of over 95%. For cooking recipes, users can save 8% keystrokes at 60% precision or 5% at 80% precision. For weather reports, keystroke savings are 2% at 70% correct suggestions or 0.8% at 80%. Finally, Jeff Dasovich of Enron can enjoy only a marginal benefit: below 1% of keystrokes are saved at 60% entirely acceptable suggestions, or 0.2% at 80% precision.

How do these performance results correlate with properties of the model and text collections? In Figure 1, we see that the mixture weights of the higher order N-gram models are greatest for the service center mails, smaller for the recipes, even smaller for the weather reports and smallest for Enron. With 50% of the mixture weights allocated to the 1-gram model, for the Enron collection the N-gram completion method can often only guess words with high prior probability. From Table 2, we can furthermore see that the entropy of the text collection is inversely proportional to the model's ability to solve the sentence completion problem. With an entropy

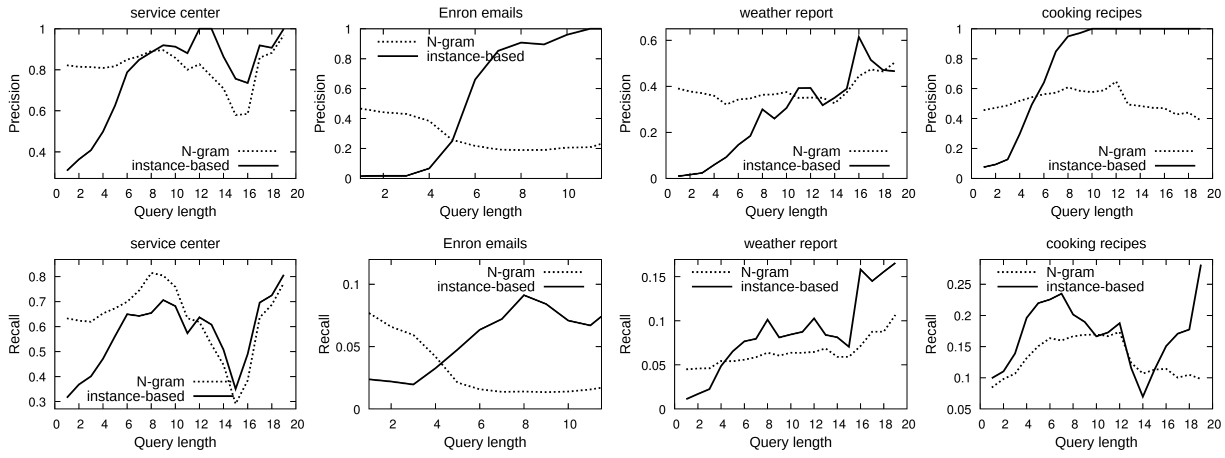


Figure 3: Precision and recall dependent on string length of initial fragment (words).

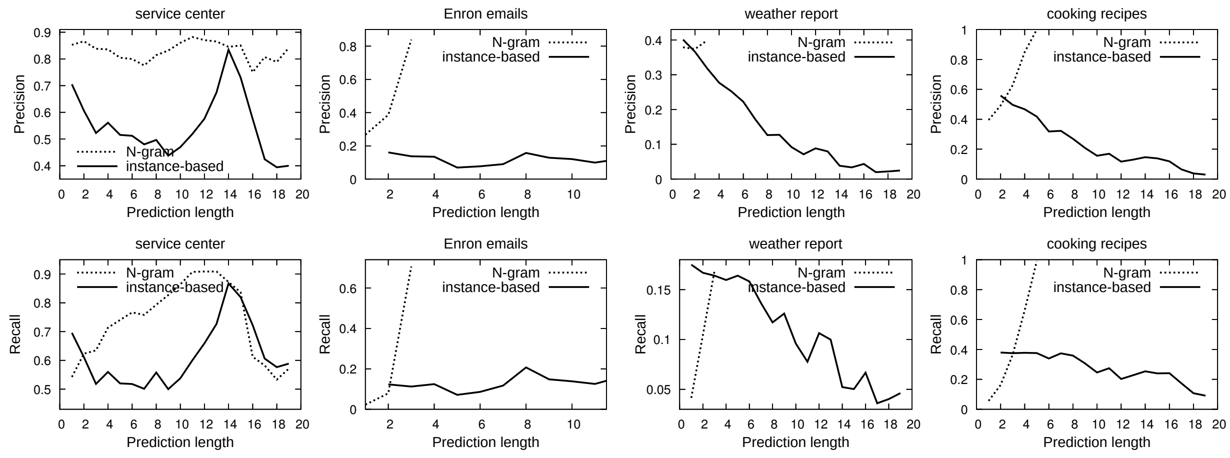


Figure 4: Precision and recall dependent on prediction string length (words).

of only 1.41, service center emails are excellently predictable; by contrast, Jeff Dasovich's personal emails have an entropy of 7.17 and are almost as unpredictable as Enron's share price.

6 Conclusion

We discussed the problem of predicting how a user will complete a sentence. We find precision (the number of suggested characters that the user has to read for every character that is accepted) and recall (the rate of keystroke savings) to be appropriate performance metrics. We developed a sentence completion method based on N-gram language models. We derived a k best Viterbi beam search decoder. Our experiments lead to the following conclusions:

(a) The N-gram based completion method has a better precision recall profile than index-based retrieval of the most similar sentence. It can be tuned to a wide range of trade-offs, a high precision can be obtained. The execution time of the Viterbi beam search decoder is in the order of few milliseconds.

(b) Whether sentence completion is helpful strongly depends on the diversity of the document collection as, for instance, measured by the entropy. For service center emails, a keystroke saving of 60% can be achieved at 85% acceptable suggestions; by contrast, only a marginal keystroke saving of 0.2% can be achieved for Jeff Dasovich's personal emails at 80% acceptable suggestions. A modest but significant benefit can be observed for thematically related documents: weather reports and cooking recipes.

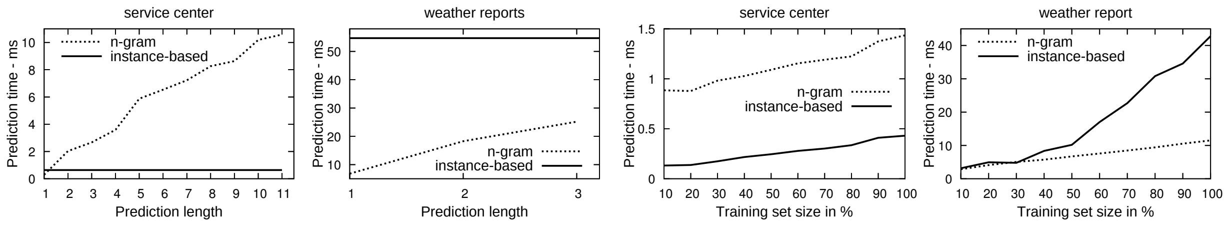


Figure 5: Prediction time dependent on prediction length in words (left) and prediction time dependent on training set size (right) for *service center* and *weather report* collections.

Acknowledgment

This work has been supported by the German Science Foundation DFG under grant SCHE540/10.

References

P. Brown, S. Della Pietra, V. Della Pietra, J. Lai, and R. Mercer. 1992. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(2):31–40.

J. Darragh and I. Witten. 1992. *The Reactive Keyboard*. Cambridge University Press.

B. Davison and H. Hirsch. 1998. Predicting sequences of user actions. In *Proceedings of the AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time Series Analysis*.

M. Debevc, B. Meyer, and R. Svecko. 1997. An adaptive short list for documents on the world wide web. In *Proceedings of the International Conference on Intelligent User Interfaces*.

G. Foster, P. Langlais, and G. Lapalme. 2002. User-friendly text prediction for translators. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

N. Garay-Vitoria and J. Abascal. 2004. A comparison of prediction techniques to enhance the communication of people with disabilities. In *Proceedings of the 8th ERCIM Workshop User Interfaces For All*.

K. Grabski and T. Scheffer. 2004. Sentence completion. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*.

N. Jacobs and H. Blockeel. 2001. The learning shell: automated macro induction. In *Proceedings of the International Conference on User Modelling*.

N. Jacobs and H. Blockeel. 2003. Sequence prediction with mixed order Markov chains. In *Proceedings of the Belgian/Dutch Conference on Artificial Intelligence*.

B. Klimt and Y. Yang. 2004. The Enron corpus: A new dataset for email classification research. In *Proceedings of the European Conference on Machine Learning*.

B. Korvemaker and R. Greiner. 2000. Predicting Unix command lines: adjusting to user patterns. In *Proceedings of the National Conference on Artificial Intelligence*.

P. Langlais, G. Foster, and G. Lapalme. 2000. Unit completion for a computer-aided translation typing system. *Machine Translation*, 15:267–294.

P. Langlais, M. Loranger, and G. Lapalme. 2002. Translators at work with transtype: Resource and evaluation. In *Proceedings of the International Conference on Language Resources and Evaluation*.

P. Langlais, G. Lapalme, and M. Loranger. 2004. Transtype: Development-evaluation cycles to boost translator's productivity. *Machine Translation (Special Issue on Embedded Machine Translation Systems*, 17(17):77–98.

T. Magnuson and S. Hunnicutt. 2002. Measuring the effectiveness of word prediction: The advantage of long-term use. Technical Report TMH-QPSR Volume 43, Speech, Music and Hearing, KTH, Stockholm, Sweden.

H. Motoda and K. Yoshida. 1997. Machine learning techniques to make computers easier to use. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*.

L. Nepveu, G. Lapalme, P. Langlais, and G. Foster. 2004. Adaptive language and translation models for interactive machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

C. Shannon. 1951. Prediction and entropy of printed english. In *Bell Systems Technical Journal, 30, 50-64*.

W. Zagler and C. Beck. 2002. FASTY - faster typing for disabled persons. In *Proceedings of the European Conference on Medical and Biological Engineering*.

Training Neural Network Language Models On Very Large Corpora [*]

Holger Schwenk and **Jean-Luc Gauvain**
LIMSI-CNRS
BP 133, 91436 Orsay cedex, FRANCE
schwenk,gauvain@limsi.fr

Abstract

During the last years there has been growing interest in using neural networks for language modeling. In contrast to the well known back-off n-gram language models, the neural network approach attempts to overcome the data sparseness problem by performing the estimation in a continuous space. This type of language model was mostly used for tasks for which only a very limited amount of in-domain training data is available.

In this paper we present new algorithms to train a neural network language model on very large text corpora. This makes possible the use of the approach in domains where several hundreds of millions words of texts are available. The neural network language model is evaluated in a state-of-the-art real-time continuous speech recognizer for French Broadcast News. Word error reductions of 0.5% absolute are reported using only a very limited amount of additional processing time.

1 Introduction

Language models play an important role in many applications like character and speech recognition, machine translation and information retrieval. Several approaches have been developed during the last decades like n-gram back-off word models (Katz, 1987), class models (Brown et al., 1992), structured language models (Chelba and Jelinek, 2000) or maximum entropy language models (Rosenfeld, 1996). To the best of our knowledge word and class n-gram back-off language models are still the dominant approach, at least in applications like large vocabulary continuous speech recognition or statistical machine translation. In many publications it has been reported that modified Kneser-Ney smoothing (Chen and Goodman, 1999) achieves the best results. All the reference back-off language models (LM) described in this paper are build with this technique, using the SRI LM toolkit (Stolcke, 2002).

The field of natural language processing has recently seen some changes by the introduction of new statistical techniques that are motivated by successful approaches from the machine learning community, in particular continuous space LMs using neural networks (Bengio and Ducharme, 2001; Bengio et al., 2003; Schwenk and Gauvain, 2002; Schwenk and Gauvain, 2004; Emami and Jelinek, 2004), Random Forest LMs (Xu and Jelinek, 2004) and Random cluster LMs (Emami and Jelinek, 2005). Usually new approaches are first verified on small tasks using a limited amount of LM training data. For instance, experiments have been performed using the Brown corpus (1.1M words), parts of the Wallstreet journal corpus (19M words) or transcriptions of acoustic training data (up to 22M words). It is much more challenging to compare the new statistical techniques to carefully optimized back-off LM trained on large amounts of data (several hundred millions words). Training may be difficult and very

[*]This work was partially financed by the European Commission under the FP6 Integrated Project TC-STAR.

time consuming and the algorithms used with several tens of millions examples may be impracticable for larger amounts. Training back-off LMs on large amounts of data is not a problem, as long as powerful machines with enough memory are available in order to calculate the word statistics. Practice has also shown that back-off LMs seem to perform very well when large amounts of training data are available and it is not clear that the above mentioned new approaches are still of benefit in this situation.

In this paper we compare the neural network language model to n-gram model with modified Kneser-Ney smoothing using LM training corpora of up to 600M words. New algorithms are presented to effectively train the neural network on such amounts of data and the necessary capacity is analyzed. The LMs are evaluated in a real-time state-of-the-art speech recognizer for French Broadcast News. Word error reductions of up to 0.5% absolute are reported.

2 Architecture of the neural network LM

The basic idea of the neural network LM is to project the word indices onto a continuous space and to use a probability estimator operating on this space (Bengio and Ducharme, 2001; Bengio et al., 2003). Since the resulting probability functions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. A neural network can be used to simultaneously learn the projection of the words onto the continuous space and to estimate the n-gram probabilities. This is still a n-gram approach, but the LM posterior probabilities are "interpolated" for any possible context of length n-1 instead of backing-off to shorter contexts.

The architecture of the neural network n-gram LM is shown in Figure 1. A standard fully-connected multi-layer perceptron is used. The inputs to the neural network are the indices of the $n-1$ previous words in the vocabulary $h_j = w_{j-n+1}, ..., w_{j-2}, w_{j-1}$ and the outputs are the posterior probabilities of *all* words of the vocabulary:

$$P(w_j = i|h_j) \qquad \forall i \in [1, N] \tag{1}$$

where N is the size of the vocabulary. The input uses the so-called 1-of-n coding, i.e., the *i-th* word of the vocabulary is coded by setting the *i-th* element of the vector to 1 and all the other elements to

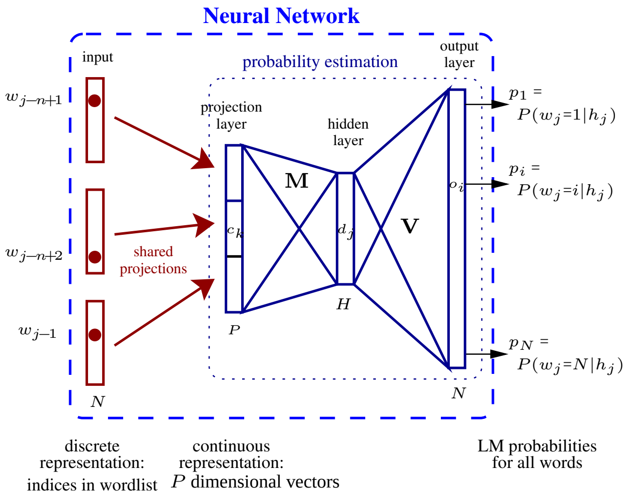


Figure 1: Architecture of the neural network language model. h_j denotes the context $w_{j-n+1}, ..., w_{j-1}$. P is the size of one projection and H and N is the size of the hidden and output layer respectively. When shortlists are used the size of the output layer is much smaller then the size of the vocabulary.

0. The *i-th* line of the $N \times P$ dimensional projection matrix corresponds to the continuous representation of the i-th word. Let us denote c_k these projections, d_j the hidden layer activities, o_i the outputs, p_i their softmax normalization, and m_{jl}, b_j, v_{ij} and k_i the hidden and output layer weights and the corresponding biases. Using these notations the neural network performs the following operations:

$$d_j = \tanh\left(\sum_l m_{jl}\, c_l + b_j\right) \tag{2}$$

$$o_i = \sum_j v_{ij}\, d_j + k_i \tag{3}$$

$$p_i = e^{o_i} \Big/ \sum_{k=1}^{N} e^{o_k} \tag{4}$$

The value of the output neuron p_i corresponds directly to the probability $P(w_j = i|h_j)$. Training is performed with the standard back-propagation algorithm minimizing the following error function:

$$E = \sum_{i=1}^{N} t_i \, log\, p_i + \beta(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2) \tag{5}$$

where t_i denotes the desired output, i.e., the probability should be 1.0 for the next word in the training sentence and 0.0 for all the other ones. The first part of this equation is the cross-entropy between the output and the target probability distributions, and the second part is a regularization term that aims to prevent the neural network from overfitting the training data (weight decay). The parameter β has to be determined experimentally.

It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities. Therefore, the neural network directly minimizes the perplexity on the training data. Note also that the gradient is back-propagated through the projection-layer, which means that the neural network learns the projection of the words onto the continuous space that is best for the probability estimation task. The complexity to calculate one probability with this basic version of the neural network LM is quite high:

$$O = (n-1) \times P \times H + H + H \times N + N \quad (6)$$

where P is the size of one projection and H and N is the size of the hidden and output layer respectively. Usual values are n=4, P=50 to 200, H=400 to 1000 and N=40k to 200k. The complexity is dominated by the large size of the output layer. In this paper the improvements described in (Schwenk, 2004) have been used:

1. *Lattice rescoring*: speech recognition is done with a standard back-off LM and a word lattice is generated. The neural network LM is then used to rescore the lattice.

2. *Shortlists*: the neural network is only used to predict the LM probabilities of a subset of the whole vocabulary.

3. *Regrouping*: all LM probabilities needed for one lattice are collected and sorted. By these means all LM probability requests with the same context h_t lead to only one forward pass through the neural network.

4. *Block mode*: several examples are propagated at once through the neural network, allowing the use of faster matrix/matrix operations.

5. *CPU optimization*: machine specific BLAS libraries are used for fast matrix and vector operations.

The idea behind shortlists is to use the neural network only to predict the s most frequent words, $s \ll |V|$, reducing by these means drastically the complexity. All words of the word list are still considered at the input of the neural network. The LM probabilities of words in the shortlist ($\hat{P}_N$) are calculated by the neural network and the LM probabilities of the remaining words ($\hat{P}_B$) are obtained from a standard 4-gram back-off LM:

$$\hat{P}(w_t|h_t) = \begin{cases} \hat{P}_N(w_t|h_t) P_S(h_t) & \text{if } w_t \in \text{shortlist} \\ \hat{P}_B(w_t|h_t) & \text{else} \end{cases} \quad (7)$$

$$P_S(h_t) = \sum_{w \in shortlist(h_t)} \hat{P}_B(w|h_t) \quad (8)$$

It can be considered that the neural network redistributes the probability mass of all the words in the shortlist. This probability mass is precalculated and stored in the data structures of the back-off LM. A back-off technique is used if the probability mass for a requested input context is not directly available.

Normally, the output of a speech recognition system is the most likely word sequence given the acoustic signal, but it is often advantageous to preserve more information for subsequent processing steps. This is usually done by generating a lattice, a graph of possible solutions where each arc corresponds to a hypothesized word with its acoustic and language model scores. In the context of this work LIMSI's standard large vocabulary continuous speech recognition decoder is used to generate lattices using a n-gram back-off LM. These lattices are then processed by a separate tool and all the LM probabilities on the arcs are replaced by those calculated by the neural network LM. During this lattice rescoring LM probabilities with the same context h_t are often requested several times on potentially different nodes in the lattice. Collecting and regrouping all these calls prevents multiple forward passes since all LM predictions for the same context are immediately available at the output.

Further improvements can be obtained by propagating several examples at once though the network, also known as bunch mode (Bilmes et al., 1997; Schwenk, 2004). In comparison to equation 2 and 3, this results in using matrix/matrix instead of matrix/vector operations which can be aggressively optimized on current CPU architectures. The Intel

Math Kernel Library was used.[1] Bunch mode is also used for training the neural network. Training of a typical network with a hidden layer with 500 nodes and a shortlist of length 2000 (about 1M parameters) take less than one hour for one epoch through four million examples on a standard PC.

3 Application to Speech Recognition

In this paper the neural network LM is evaluated in a real-time speech recognizer for French Broadcast News. This is a very challenging task since the incorporation of the neural network LM into the speech recognizer must be very effective due to the time constraints. The speech recognizer itself runs in 0.95xRT[2] and the neural network in less than 0.05xRT. The compute platform is an Intel Pentium 4 extreme (3.2GHz, 4GB RAM) running Fedora Core 2 with hyper-threading.

The acoustic model uses tied-state position-dependent triphones trained on about 190 hours of Broadcast News data. The speech features consist of 39 cepstral parameters derived from a Mel frequency spectrum estimated on the 0-8kHz band (or 0-3.8kHz for telephone data) every 10ms. These cepstral coefficients are normalized on a segment cluster basis using cepstral mean removal and variance normalization. The feature vectors are linearly transformed (MLLT) to better fit the diagonal covariance Gaussians used for acoustic modeling.

Decoding is performed in two passes. The first fast pass generates an initial hypothesis, followed by acoustic model adaptation (CMLLR and MLLR) and a second decode pass using the adapted models. Each pass generates a word lattice which is expanded with a 4-gram LM. The best solution is then extracted using pronunciation probabilities and consensus decoding. Both passes use very tight pruning thresholds, especially for the first pass, and fast Gaussian computation based on Gaussian short lists. For the final decoding pass, the acoustic models include 23k position-dependent triphones with 12k tied states, obtained using a divisive decision tree based clustering algorithm with a 35 base phone set.

[1]http://www.intel.com/software/products/mkl/

[2]In speech recognition, processing time is measured in multiples of the length of the speech signal, the real time factor xRT. For a speech signal of 2h, a processing time of 0.5xRT corresponds to 1h of calculation.

The system is described in more detail in (Gauvain et al., 2005).

The neural network LM is used in the last pass to rescore the lattices. A short-list of length 8192 was used in order to fulfill the constraints on the processing time (the complexity of the neural network to calculate a LM probability is almost linear with the length of the short-list). This gives a coverage of about 85% when rescoring the lattices, i.e. the percentage of LM requests that are actually performed by the neural network.

3.1 Language model training data

The following resources have been used for language modeling:

- Transcriptions of the acoustic training data (4.0M words)
- Commercial transcriptions (88.5M words)
- Newspaper texts (508M words)
- WEB data (13.6M words)

First a language model was built for each corpus using modified Kneser-Ney smoothing as implemented in the SRI LM toolkit (Stolcke, 2002). The individual LMs were then interpolated and merged together. An EM procedure was used to determine the coefficients that minimize the perplexity on the development data. Table 1 summarizes the characteristics of the individual text corpora.

corpus	**#words**	**Perpl.**	**Coeffs.**
Acoustic transcr.	4M	107.4	0.43
Commercial transcr.	88.5M	137.8	0.14
Newspaper texts	508M	103.0	0.35
WEB texts	13.6M	136.7	0.08
All interpolated	614M	70.2	-

Table 1: Characteristics of the text corpora (number of words, perplexity on the development corpus and interpolation coefficients)

Although the detailed transcriptions of the audio data represent only a small fraction of the available data, they get an interpolation coefficient of 0.43. This shows clearly that they are the most appropriate text source for the task. The commercial transcripts, the newspaper and WEB texts reflect less well the speaking style of broadcast news, but this is to some extent counterbalanced by the large amount of data. One could say that these texts are helpful to learn the general grammar of the language. The word list includes 65301 words and the OOV rate is 0.95% on a development set of 158k words.

3.2 Training on in-domain data only

Following the above discussion, it seems natural to first train a neural network LM on the transcriptions of the acoustic data only. The architecture of the neural network is as follows: a continuous word representation of dimension 50, one hidden layer with 500 neurons and an output layer limited to the 8192 most frequent words. This results in 3.2M parameters for the continuous representation of the words and about 4.2M parameters for the second part of the neural network that estimates the probabilities. The network is trained using standard stochastic back-propagation.[3] The learning rate was set to 0.005 with an exponential decay and the regularization term is weighted with 0.00003. Note that fast training of neural networks with more than 4M parameters on 4M examples is already a challenge. The same fast algorithms as described in (Schwenk, 2004) were used. Apparent convergence is obtained after about 40 epochs though the training data, each one taking 2h40 on standard PC equipped with two Intel Xeon 2.8GHz CPUs.

The neural network LM alone achieves a perplexity of 103.0 which is only a 4% relative reduction with respect to the back-off LM (107.4, see Table 1). If this neural network LM is interpolated with the back-off LM trained on the whole training set the perplexity decreases from 70.2 to 67.6. Despite this small improvements in perplexity a notable word error reduction was obtained from 14.24% to 14.02%, with the lattice rescoring taking less than 0.05xRT. In the following sections, it is shown that larger improvements can be obtained by training the neural network on more data.

3.3 Adding selected data

Training the neural network LM with stochastic back-propagation on all the available text corpora would take quite a long time. The estimated time for one training epoch with the 88M words of commercial transcriptions is 58h, and more than 12 days if all the 508M words of newspaper texts were used. This is of course not very practicable. One solution to this problem is to select a subset of the data that seems to be most useful for the task. This was done by selecting six month of the commercial transcriptions that minimize the perplexity on the development set. This gives a total of 22M words and the training time is about 14h per epoch.

One can ask if the capacity of the neural network should be augmented in order to deal with the increased number of examples. Experiments with hidden layer sizes from 400 to 1000 neurons have been performed (see Table 2).

size	**400**	**500**	**600**	**1000***
Tr. time	11h20	13h50	16h15	11+16h
Px alone	100.5	100.1	99.5	94.5
interpol.	68.3	68.3	68.2	68.0
Werr	13.99%	13.97%	13.96%	13.92%

* Interpolation of networks with 400 and 600 hidden units.

Table 2: Performance for a neural network LM and training time per epoch as a function of the size of the hidden layer (fixed 6 months subset of commercial transcripts).

Although there is a small decrease in perplexity and word error when increasing the dimension of the hidden layer, this is at the expense of a higher processing time. The training and recognition time are in fact almost linear to the size of the hidden layer. An alternative approach to augment the capacity of the neural network is to modify the dimension of the continuous representation of the words (in the range 50 to 150). The idea behind this is that the probability estimation may be easier in a higher dimensional space (instead of augmenting the capacity of the non-linear probability estimator itself). This is similar in spirit to the theory behind support vector machines (Vapnik, 1998).

Increasing the dimension of the projection layer has several advantages as can be seen from the Figure 2. First, the perplexity and word error rates are lower than those obtained when the size of the

[3]The weights are updated after each example.

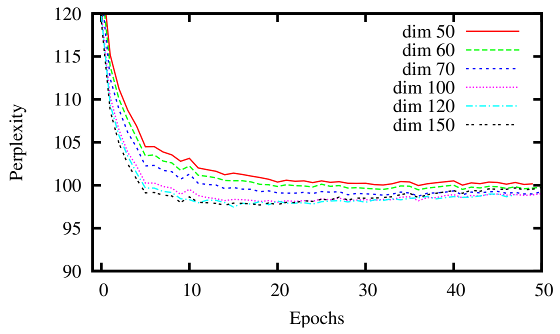


Figure 2: Perplexity in function of the size of the continuous word representation (500 hidden units, fixed 6 months subset of commercial transcripts).

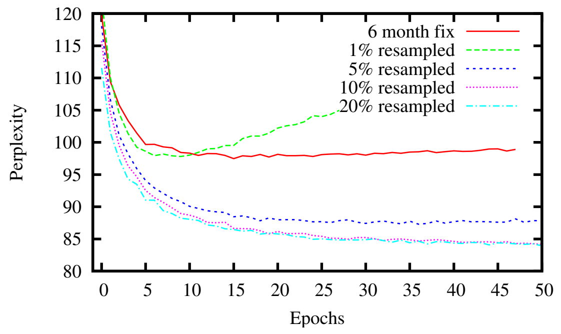


Figure 3: Perplexity when resampling different random subsets of the commercial transcriptions. (word representation of dimension 120, 500 hidden units)

hidden layer is increased. Second, convergence is faster: the best result is obtained after about 15 epochs while up to 40 are needed with large hidden layers. Finally, increasing the size of the continuous word representation has only a small effect on the training and recognition complexity of the neural network[4] since most of the calculation is done to propagate and learn the connections between the hidden and the output layer (see equation 6). The best result was obtained with a 120 dimensional continuous word representation. The perplexity is 67.9 after interpolation with the back-off LM and the word error rate is 13.88%.

3.4 Training on all available data

In this section an algorithm is proposed for training the neural network on arbitrary large training corpora. The basic idea is quite simple: instead of performing several epochs over the whole training data, a different small random subset is used at each epoch. This procedure has several advantages:

- There is no limit on the amount of training data,
- After some epochs, it is likely that all the training examples have been seen at least once,
- Changing the examples after each epoch adds noise to the training procedure. This potentially increases the generalization performance.

This algorithm is summarized in figure 4. The parameters of this algorithm are the size of the random subsets that are used at each epoch. We chose to always use the full corpus of transcriptions of the acoustic data since this is the most appropriate data for the task. Experiments with different random subsets of the commercial transcriptions and the newspaper texts have been performed (see Figure 3 and 5). In all cases the same neural network architecture was used, i.e a 120 dimensional continuous word representation and 500 hidden units. Some experiments with larger hidden units showed basically the same convergence behavior. The learning rate was again set to 0.005, but with a slower exponential decay.

First of all it can be seen from Figure 3 that the results are better when using random subsets instead of a fixed selection of 6 months, although each random subset is actually smaller (for instance a total of 12.5M examples for a subset of 10%). Best results were obtained when taking 10% of the commercial

```
Repeat
  Select training data:
    – Use all acoustic transcriptions (4M words)
    – Extract random subset of examples
      from the large corpora
    – Shuffle data
  + Train network for one epoch
    (performing weight updates after each example)
  + Test performance on development data
Until convergence
```

Figure 4: Training algorithm for large corpora

[4]14h20 for P=120 and H=500.

	Back-off LM	**Neural Network LM**				
Training data [#words]	600M	4M	22M	92.5M*	600M*	
Training time [h/epoch]	-	2h40	14h	9h40	12h	3 × 12h
Perplexity (NN LM alone)	-	103.0	97.5	84.0	80.0	76.5
Perplexity (interpolated LMs)	70.2	67.6	67.9	66.7	66.5	65.9
Word error rate (interpolated LMs)	**14.24%**	14.02%	13.88%	13.81%	**13.75%**	**13.61%**

* By resampling different random parts at the beginning of each epoch.

Table 3: Comparison of the back-off and the neural network LM using different amounts of training data. The perplexities are given for the neural network LM alone and interpolated with the back-off LM trained on all the data. The last column corresponds to three interpolated neural network LMs.

transcriptions. The perplexity is 66.7 after interpolation with the back-off LM and the word error rate is 13.81% (see summary in Table 3). Larger subsets of the commercial transcriptions lead to slower training, but don't give better results.

Encouraged by these results, we also included the 508M words of newspaper texts in the training data. The size of the random subsets were chosen in order to use between 4 and 9M words of each corpus. Figure 5 summarizes the results. There seems to be no obvious benefit from resampling large subsets of the individual corpora. We choose to resample 10% of the commercial transcriptions and 1% of the newspaper texts.

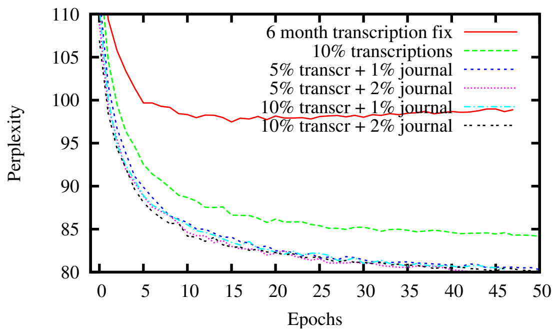


Figure 5: Perplexity when resampling different random subsets of the commercial transcriptions and the newspaper texts.

Table 3 summarizes the results of the different neural network LMs. It can be clearly seen that the perplexity of the neural network LM alone decreases significantly with the amount of training data used. The perplexity after interpolation with the back-off LM changes only by a small amount, but there is a notable improvement in word error rate. This is another experimental evidence that the perplexity of a LM is not directly related to the word error rate.

The best neural network LM achieves a word error reduction of 0.5% absolute with respect to the carefully tuned back-off LM (14.24% → 13.75%). The additional processing time needed to rescore the lattices is less than 0.05xRT. This is a significant improvement, in particular for a fast real-time continuous speech recognition system. When more processing time is available a word error rate of 13.61% can be achieved by interpolating three neural networks together (in 0.14xRT).

3.5 Using a better speech recognizer

The experimental results have also been validated using a second speech recognizer running in about 7xRT. This systems differs from the real-time recognizer by a larger 200k word-list, additional acoustic model adaptation passes and less pruning. Details are described in (Gauvain et al., 2005). The word error rate of the reference system using a back-off LM is 10.74%. This can be reduced to 10.51% using a neural network LM trained on the fine transcriptions only and to 10.20% when the neural network LM is trained on all data using the described resampling approach. Lattice rescoring takes about 0.2xRT.

4 Conclusions and future work

Neural network language models are becoming a serious alternative to the widely used back-off language models. Consistent improvements in perplexity and word error rate have been reported (Bengio et al., 2003; Schwenk and Gauvain, 2004; Schwenk and Gauvain, 2005; Emami and Jelinek, 2004). In these works, the amount of training data was however limited to a maximum of 20M words due to the high complexity of the training algorithm.

In this paper new techniques have been described to train neural network language models on large amounts of text corpora (up to 600M words). The evaluation with a state-of-the-art speech recognition system for French Broadcast News showed a significant word error reduction of 0.5% absolute. The neural network LMs is incorporated into the speech recognizer by rescoring lattices. This is done in less than 0.05xRT.

Several extensions of the learning algorithm itself are promising. We are in particular interested in smarter ways to select different subsets from the large corpus at each epoch (instead of a random choice). One possibility would be to use active learning, i.e. focusing on examples that are most useful to decrease the perplexity. One could also imagine to associate a probability to each training example and to use these probabilities to weight the random sampling. These probabilities would be updated after each epoch. This is similar to boosting techniques (Freund, 1995) which build sequentially classifiers that focus on examples wrongly classified by the preceding one.

5 Acknowledgment

The authors would like to thank Yoshua Bengio for fruitful discussions and helpful comments. The authors would like to recognize the contributions of G. Adda, M. Adda and L. Lamel for their involvement in the development of the speech recognition systems on top of which this work is based.

References

Yoshua Bengio and Rejean Ducharme. 2001. A neural probabilistic language model. In *NIPS*, volume 13.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(2):1137–1155.

Jeff Bilmes, Krste Asanovic, Chee whye Chin, and Jim Demmel. 1997. Using phipac to speed error backpropagation learning. In *ICASSP*, pages V:4153–4156.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–470.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 13(4):283–332.

Stanley F. Chen and Joshua T. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Ahmad Emami and Frederick Jelinek. 2004. Exact training of a neural syntactic language model. In *ICASSP*, pages I:245–248.

Ahmad Emami and Frederick Jelinek. 2005. Random clusterings for language modeling. In *ICASSP*, pages I:581–584.

Yoav Freund. 1995. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285.

Jean-Luc Gauvain, Gilles Adda, Martine Adda-Decker, Alexandre Allauzen, Veronique Gendner, Lori Lamel, and Holger Schwenk. 2005. Where are we in transcribing BN french? In *Eurospeech*.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on ASSP*, 35(3):400–401.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech & Language*, 10(3):187–228.

Holger Schwenk and Jean-Luc Gauvain. 2002. Connectionist language modeling for large vocabulary continuous speech recognition. In *ICASSP*, pages I: 765–768.

Holger Schwenk and Jean-Luc Gauvain. 2004. Neural network language models for conversational speech recognition. In *ICSLP*, pages 1215–1218.

Holger Schwenk and Jean-Luc Gauvain. 2005. Building continuous space language models for transcribing european languages. In *Eurospeech*.

Holger Schwenk. 2004. Efficient training of large neural networks for language modeling. In *IJCNN*, pages 3059–3062.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*, pages II: 901–904.

Vladimir Vapnik. 1998. *Statistical Learning Theory*. Wiley, New York.

Peng Xu and Frederick Jelinek. 2004. Random forest in language modeling. In *EMNLP*, pages 325–332.

Minimum Sample Risk Methods for Language Modeling[1]

Jianfeng Gao
Microsoft Research Asia
jfgao@microsoft.com

Hao Yu, Wei Yuan
Shanghai Jiaotong Univ., China

Peng Xu
John Hopkins Univ., U.S.A.
xp@clsp.jhu.edu

Abstract

This paper proposes a new discriminative training method, called *minimum sample risk* (MSR), of estimating parameters of language models for text input. While most existing discriminative training methods use a loss function that can be optimized easily but approaches only approximately to the objective of minimum error rate, MSR minimizes the training error directly using a heuristic training procedure. Evaluations on the task of Japanese text input show that MSR can handle a large number of features and training samples; it significantly outperforms a regular trigram model trained using maximum likelihood estimation, and it also outperforms the two widely applied discriminative methods, the boosting and the perceptron algorithms, by a small but statistically significant margin.

1 Introduction

Language modeling (LM) is fundamental to a wide range of applications, such as speech recognition and Asian language text input (Jelinek 1997; Gao et al. 2002). The traditional approach uses a parametric model with *maximum likelihood estimation* (MLE), usually with smoothing methods to deal with data sparseness problems. This approach is optimal under the assumption that the true distribution of data on which the parametric model is based is known. Unfortunately, such an assumption rarely holds in realistic applications.

An alternative approach to LM is based on the framework of discriminative training, which uses a much weaker assumption that training and test data are generated from the same distribution but the form of the distribution is unknown. Unlike the traditional approach that maximizes the function (i.e. 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. So, they potentially lead to better solutions. However, the error rate of a finite set of training samples is usually a step function of model parameters, and cannot be easily minimized. To address this problem, previous research has concentrated on the development of a loss function that approximates the exact error rate and can be easily optimized. Though these methods (e.g. the boosting method) have theoretically appealing properties, such as convergence and bounded generalization error, we argue that the approximated loss function may prevent them from attaining the original objective of minimizing the error rate.

In this paper we present a new estimation procedure for LM, called *minimum sample risk* (MSR). It differs from most existing discriminative training methods in that instead of searching on an approximated loss function, MSR employs a simple heuristic training algorithm that minimizes the error rate on training samples directly. MSR operates like a multidimensional function optimization algorithm: first, it selects a subset of features that are the most effective among all candidate features. The parameters of the model are then optimized iteratively: in each iteration, only the parameter of one feature is adjusted. Both feature selection and parameter optimization are based on the criterion of minimizing the error on training samples. Our evaluation on the task of Japanese text input shows that MSR achieves more than 20% error rate reduction over MLE on two newswire data sets, and it also outperforms the other two widely applied discriminative methods, the boosting method and the perceptron algorithm, by a small but statistically significant margin.

Although it has not been proved in theory that MSR is always *robust*, our experiments of cross-domain LM adaptation show that it is. MSR can effectively adapt a model trained on one domain to

[1] The work was done while the second, third and fourth authors were visiting Microsoft Research Asia. Thanks to Hisami Suzuki for her valuable comments.

different domains. It outperforms the traditional LM adaptation method significantly, and achieves at least comparable or slightly better results to the boosting method and the perceptron algorithm.

2 IME Task and LM

This paper studies LM on the task of Asian language (e.g. Chinese or Japanese) text input. This is the 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 *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. Current IME systems make about 5-15% CER in conversion of real data in a wide variety of domains (e.g. Gao et al. 2002).

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 since the phonetic string is inputted by users. Moreover, if we do not take into account typing errors, it is reasonable to assume a unique mapping from W and A in IME, i.e. $P(A \mid W) = 1$. So the decision of Equation (1) depends solely upon $P(W)$, making IME a more *direct* evaluation test bed for LM than speech recognition. Another advantage is that it is easy to convert W to A (for Chinese and Japanese), which enables us to obtain a large number of training data for discriminative learning, as described later.

The values of $P(W)$ in Equation (1) are traditionally calculated by MLE: the optimal model parameters $\boldsymbol{\lambda}^*$ are chosen in such a way that $P(W|\boldsymbol{\lambda}^*)$ is maximized on training data. The arguments in favor of MLE are based on the assumption that the form of the underlying distributions is known, and that only the values of the parameters characterizing those distributions are unknown. In using MLE for LM, one always assumes a multinomial distribution of language. For example, a trigram model makes the assumption that the next word is predicted depending only on two preceding words. However, there are many cases in natural language where words over an arbitrary distance can be related. MLE is therefore not optimal because the assumed model form is incorrect.

What are the best estimators when the model is known to be false then? In IME, we can tackle this question empirically. Best IME systems achieve the least CER. Therefore, the best estimators are those which minimize the expected error rate on unseen test data. Since the distribution of test data is unknown, we can approximately minimize the error rate on some given training data (Vapnik 1999). Toward this end, we have developed a very simple heuristic training procedure called *minimum sample risk*, as presented in the next section.

3 Minimum Sample Risk

3.1 Problem Definition

We follow the general framework of linear discriminant models described in (Duda et al. 2001). In the rest of the paper we use the following notation, adapted from Collins (2002).

• 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 $i = 1…M$, 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 $d = 0…D$. 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 $\mathbf{f}(W) = [f_0(W), f_1(W), …, f_D(W)]^T$. Without loss of generality, $f_0(W)$ 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 $d = 1…D$) 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, $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, …, \lambda_D]$. The score of a word string W can be written as

$$Score(W, \boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d=0}^{D} \lambda_d f_d(W) \cdot \quad (2)$$

The decision rule of Equation (1) is rewritten as

$$W^*(A, \boldsymbol{\lambda}) = \arg\max_{W \in \mathbf{GEN}(A)} Score(W, \boldsymbol{\lambda}) \cdot \quad (3)$$

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 $\mathrm{Er}(W^R,W)$ (i.e. the string edit distance function in our case). We call the sum of error counts over the training samples *sample risk*. Our goal is to minimize the sample risk while searching for the parameters as defined in Equation (4), hence the name *minimum sample risk* (MSR). W_i^* in Equation (4) is determined by Equation (3),

$$\boldsymbol{\lambda}_{MSR} \overset{def}{=} \arg\min_{\boldsymbol{\lambda}} \sum_{i=1\ldots M} \mathrm{Er}(W_i^R, W_i^*(A_i, \boldsymbol{\lambda})) \cdot \quad (4)$$

We first present the basic MSR training algorithm, and then the two improvements we made.

3.2 Training Algorithm

The MSR training algorithm is cast as a multidimensional function optimization approach (Press et al. 1992): taking the feature vector as a set of directions; the first direction (i.e. feature) is selected and the objective function (i.e. sample risk) is minimized along that direction using a *line search*; then from there along the second direction to its minimum, and so on, cycling through the whole set of directions as many times as necessary, until the objective function stops decreasing.

This simple method can work properly under two assumptions. First, there exists an implementation of line search that optimizes the function along one direction efficiently. Second, the number of candidate features is not too large, and these features are not highly correlated. However, neither of the assumptions holds in our case. First of all, Er(.) in Equation (4) is a step function of $\boldsymbol{\lambda}$, thus cannot be optimized directly by regular gradient-based procedures – a grid search has to be used instead. However, there are problems with simple grid search: using a large grid could miss the optimal solution whereas using a fine-grained grid would lead to a very slow algorithm. Secondly, in the case of LM, there are millions of candidate features, some of which are highly correlated. We address these issues respectively in the next two subsections.

3.3 Grid Line Search

Our implementation of a grid search is a modified version of that proposed in (Och 2003). The modifications are made to deal with the efficiency issue due to the fact that there is a very large number of features and training samples in our task, compared to only 8 features used in (Och 2003). Unlike a simple grid search where the intervals between any two adjacent grids are equal and fixed, we determine for each feature a sequence of grids with differently sized intervals, each corresponding to a different value of sample risk.

As shown in Equation (4), the loss function (i.e. sample risk) over all training samples is the sum of the loss function (i.e. Er(.)) of each training sample. Therefore, in what follows, we begin with a discussion on minimizing Er(.) of a training sample using the line search.

Let $\boldsymbol{\lambda}$ be the current model parameter vector, and f_d be the selected feature. The line search aims to find the optimal parameter λ_d^* so as to minimize Er(.). For a training sample (A, W^R), the score of each candidate word string $W \in \mathbf{GEN}(A)$, as in Equation (2), can be decomposed into two terms:

$$Score(W, \boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d'=0 \vee d' \neq d}^{D} \lambda_{d'} f_{d'}(W) + \lambda_d f_d(W),$$

where the first term on the right hand side does not change with λ_d. Note that if several candidate word strings have the same feature value $f_d(W)$, their relative rank will remain the same for any λ_d. Since $f_d(W)$ takes integer values in our case ($f_d(W)$ is the count of a particular *n*-gram in W), we can group the candidates using $f_d(W)$ so that candidates in each group have the same value of $f_d(W)$. In each group, we define the candidate with the highest value of

$$\sum\nolimits_{d'=0 \vee d' \neq d}^{D} \lambda_{d'} f_{d'}(W)$$

as the *active* candidate of the group because no matter what value λ_d takes, only this candidate could be selected according to Equation (3).

Now, we reduce $\mathbf{GEN}(A)$ to a much smaller list of active candidates. We can find a set of intervals for λ_d, within each of which a particular active candidate will be selected as W^*. We can compute the Er(.) value of that candidate as the Er(.) value for the corresponding interval. As a result, for each training sample, we obtain a sequence of intervals and their corresponding Er(.) values. The optimal value λ_d^* can then be found by traversing the sequence and taking the midpoint of the interval with the lowest Er(.) value.

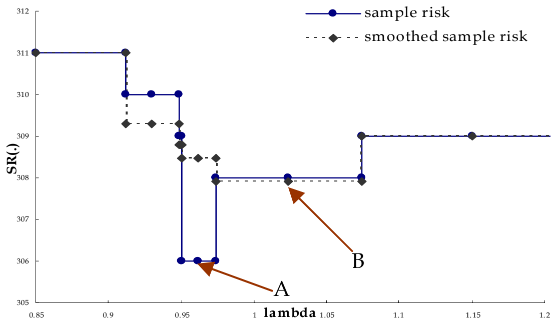


Figure 1. Examples of line search.

This process can be extended to the whole training set as follows. By merging the sequence of intervals of each training sample in the training set, we obtain a global sequence of intervals as well as their corresponding sample risk. We can then find the optimal value λ_d^* as well as the minimal sample risk by traversing the global interval sequence. An example is shown in Figure 1.

The line search can be unstable, however. In some cases when some of the intervals are very narrow (e.g. the interval A in Figure 1), moving the optimal value λ_d^* slightly can lead to much larger sample risk. Intuitively, we prefer a stable solution which is also known as a robust solution (with even slightly higher sample risk, e.g. the interval B in Figure 1). Following Quirk et al. (2004), we evaluate each interval in the sequence by its corresponding *smoothed sample risk*. Let λ be the midpoint of an interval and $\text{SR}(\lambda)$ be the corresponding sample risk of the interval. The smoothed sample risk of the interval is defined as

$$\int_{\lambda-b}^{\lambda+b} \text{SR}(\lambda)\, d\lambda$$

where b is a smoothing factor whose value is determined empirically (0.06 in our experiments). As shown in Figure 1, a more stable interval B is selected according to the smoothed sample risk.

In addition to reducing **GEN**(A) to an active candidate list described above, the efficiency of the line search can be further improved. We find that the line search only needs to traverse a small subset of training samples because the distribution of features among training samples are very sparse. Therefore, we built an inverted index that lists for each feature all training samples that contain it. As will be shown in Section 4.2, the line search is very efficient even for a large training set with millions of candidate features.

3.4 Feature Subset Selection

This section describes our method of selecting among millions of features a small subset of highly effective features for MSR learning. Reducing the number of features is essential for two reasons: to reduce computational complexity and to ensure the generalization property of the linear model. A large number of features lead to a large number of parameters of the resulting linear model, as described in Section 3.1. For a limited number of training samples, keeping the number of features sufficiently small should lead to a simpler model that is less likely to overfit to the training data.

The first step of our feature selection algorithm treats the features *independently*. The effectiveness of a feature is measured in terms of the reduction of the sample risk on top of the base feature f_0. Formally, let $\text{SR}(f_0)$ be the sample risk of using the base feature only, and $\text{SR}(f_0 + \lambda_d f_d)$ be the sample risk of using both f_0 and f_d and the parameter λ_d that has been optimized using the line search. Then the effectiveness of f_d, denoted by $E(f_d)$, is given by

$$E(f_d) = \frac{\text{SR}(f_0) - \text{SR}(f_0 + \lambda_d f_d)}{\max\limits_{f_i, i=1\ldots D}(\text{SR}(f_0) - \text{SR}(f_0 + \lambda_i f_i))}, \qquad (5)$$

where the denominator is a normalization term to ensure that $E(f) \in [0, 1]$.

The feature selection procedure can be stated as follows: The value of $E(.)$ is computed according to Equation (5) for each of the candidate features. Features are then ranked in the order of descending values of $E(.)$. The top l features are selected to form the feature vector in the linear model.

Treating features independently has the advantage of computational simplicity, but may not be effective for features with high correlation. For instance, although two features may carry rich discriminative information when treated separately, there may be very little gain if they are combined in a feature vector, because of the high correlation between them. Therefore, in what follows, we describe a technique of incorporating correlation information in the feature selection criterion.

Let x_{md}, $m = 1\ldots M$ and $d = 1\ldots D$, be a Boolean value: $x_{md} = 1$ if the sample risk reduction of using the d-th feature on the m-th training sample, computed by Equation (5), is larger than zero, and 0 otherwise. The cross correlation coefficient between two features f_i and f_j is estimated as

$$C(i,j) = \frac{\sum_{m=1}^{M} x_{mi} x_{mj}}{\sqrt{\sum_{m=1}^{M} x_{mi}^2 \sum_{m=1}^{M} x_{mj}^2}}. \qquad (6)$$

It can be shown that $C(i, j) \in [0, 1]$. Now, similar to (Theodoridis and Koutroumbas 2003), the feature selection procedure consists of the following steps, where f_i denotes any selected feature and f_j denotes any candidate feature to be selected.

Step 1. For each of the candidate features (f_d, for $d = 1\ldots D$), compute the value of $E(f)$ according to Equation (5). Rank them in a descending order and choose the one with the highest $E(.)$ value. Let us denote this feature as f_1.

Step 2. To select the second feature, compute the cross correlation coefficient between the selected feature f_1 and each of the remaining M-1 features, according to Equation (6).

Step 3. Select the second feature f according to

$$j^* = \arg\max_{j=2\ldots D} \{\alpha E(f_j) - (1-\alpha)C(1,j)\}$$

where α is the weight that determines the relative importance we give to the two terms. The value of α is optimized on held-out data (0.8 in our experiments). This means that for the selection of the second feature, we take into account not only its impact of reducing the sample risk but also the correlation with the previously selected feature. It is expected that choosing features with less correlation gives better sample risk minimization.

Step 4. Select k-th features, $k = 3\ldots K$, according to

$$j^* = \arg\max_{j} \left\{ \alpha E(f_j) - \frac{1-\alpha}{k-1} \sum_{i=1}^{k-1} C(i,j) \right\} \qquad (7)$$

That is, we select the next feature by taking into account its average correlation with all previously selected features. The optimal number of features, l, is determined on held-out data.

Similarly to the case of line search, we need to deal with the efficiency issue in the feature selection method. As shown in Equation (7), the estimates of $E(.)$ and $C(.)$ need to be computed. Let D and K ($K << D$) be the number of all candidate features and the number of features in the resulting model, respectively. According to the feature selection method described above, we need to estimate $E(.)$ for each of the D candidate features only once in Step 1. This is not very costly due to the efficiency of our line search algorithm. Unlike the case of $E(.)$, $O(K \times D)$ estimates of $C(.)$ are required in Step 4. This is computationally expensive even for a medium-sized K. Therefore, every time a new feature is selected (in Step 4), we only estimate the value of $C(.)$ between each of the selected features and each of the top N remaining features with the highest value of $E(.)$. This reduces the number of estimates of $C(.)$ to $O(K \times N)$. In our experiments we set $N = 1000$, much smaller than D. This reduces the computational cost significantly without producing any noticeable quality loss in the resulting model.

The MSR algorithm used in our experiments is summarized in Figure 2. It consists of feature selection (line 2) and optimization (lines 3 - 5) steps.

```
1  Set $\lambda_0 = 1$ and $\lambda_d = 0$ for $d=1\ldots D$
2  Rank all features and select the top $K$ features, using
   the feature selection method described in Section 3.4
3  For $t = 1\ldots T$ (T= total number of iterations)
4      For each $k = 1\ldots K$
5          Update the parameter of $f_k$ using line search.
```

Figure 2: The MSR algorithm

4 Evaluation

4.1 Settings

We evaluated MSR on the task of Japanese IME. Two newspaper corpora are used as training and test data: Nikkei and Yomiuri Newspapers. Both corpora have been pre-word-segmented using a lexicon containing 167,107 entries. A 5,000-sentence subset of the Yomiuri Newspaper corpus was used as held-out data (e.g. to determine learning rate, number of iterations and features etc.). We tested our models on another 5,000-sentence subset of the Yomiuri Newspaper corpus.

We used an 80,000-sentence subset of the Nikkei Newspaper corpus as the training set. For each A, we produced a word lattice using the baseline system described in (Gao et al. 2002), which uses a word trigram model trained via MLE on anther 400,000-sentence subset of the Nikkei Newspaper corpus. The two subsets do not overlap so as to simulate the case where unseen phonetic symbol strings are converted by the baseline system. For efficiency, we kept for each training sample the best 20 hypotheses in its candidate conversion set **GEN**(A) for discriminative training. The oracle best hypothesis, which gives the minimum number of errors, was used as the reference transcript of A.

	Model	CER (%)	% over MLE
1.	**MLE**	3.70	--
2.	**MSR (*K*=2000)**	**2.95**	**20.9**
3.	**Boosting**	3.06	18.0
4.	**Perceptron**	3.07	17.8

Table 1. Comparison of CER results.

4.2 Results

We used unigrams and bigrams that occurred more than once in the training set as features. We did not use trigram features because they did not result in a significant improvement in our pilot study. The total number of candidate features we used was around 860,000.

Our main experimental results are shown in Table 1. Row 1 is our baseline result using the word trigram model. Notice that the result is much better than the state-of-the-art performance currently available in the marketplace (e.g. Gao et al. 2002), presumably due to the large amount of training data we used, and to the similarity between the training and the test data. Row 2 is the result of the model trained using the MSR algorithm described in Section 3. We also compared the MSR algorithm to two of the state-of-the-art discriminative training methods: **Boosting** in Row 3 is an implementation of the improved algorithm for the boosting loss function proposed in (Collins 2000), and **Perceptron** in Row 4 is an implementation of the averaged perceptron algorithm described in (Collins 2002).

We see that all discriminative training methods outperform MLE significantly (p-value < 0.01). In particular, MSR outperforms MLE by more than 20% CER reduction. Notice that we used only unigram and bigram features that have been included in the baseline trigram model, so the improvement is solely attributed to the high performance of MSR. We also find that MSR outperforms the perceptron and boosting methods by a small but statistically significant margin.

The MSR algorithm is also very efficient: using a subset of 20,000 features, it takes less than 20 minutes to converge on an XEON(TM) MP 1.90GHz machine. It is as efficient as the perceptron algorithm and slightly faster than the boosting method.

4.3 Robustness Issues

Most theorems that justify the robustness of discriminative training algorithms concern two questions. First, is there a guarantee that a given algorithm converges even if the training samples are not linearly separable? This is called the *convergence* problem. Second, how well is the training error reduction preserved when the algorithm is applied to unseen test samples? This is called the *generalization* problem. Though we currently cannot give a theoretical justification, we present empirical evidence here for the robustness of the MSR approach.

As Vapnik (1999) pointed out, the most robust linear models are the ones that achieve the least training errors with the least number of features. Therefore, the robustness of the MSR algorithm are mainly affected by the feature selection method. To verify this, we created four different subsets of features using different settings of the feature selection method described in Section 3.4. We selected different numbers of features (i.e. 500 and 2000) with and without taking into account the correlation between features (i.e. α in Equation (7) is set to 0.8 and 1, respectively). For each of the four feature subsets, we used the MSR algorithm to generate a set of models. The CER curves of these models on training and test data sets are shown in Figures 3 and 4, respectively.

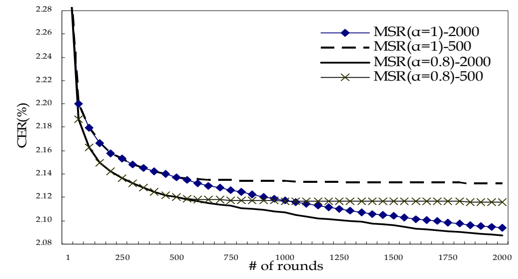


Figure 3. Training error curves of the MSR algorithm

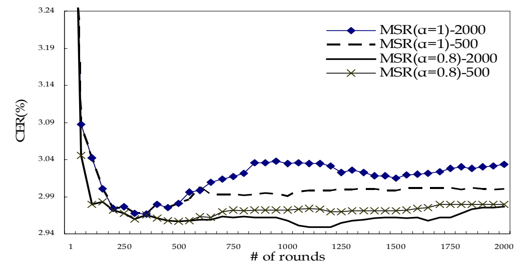


Figure 4. Test error curves of the MSR algorithm

The results reveal several facts. First, the convergence properties of MSR are shown in Figure 3 where in all cases, training errors drop consistently with more iterations. Secondly, as expected, using more features leads to overfitting, For example, MSR(α =1)-2000 makes fewer errors than MSR(α =1)-500 on training data but more errors on test data. Finally, taking into account the correlation between features (e.g. α = 0.8 in Equation (7)) results in a better subset of features that lead to not only fewer training errors, as shown in Figure 3, but also better generalization properties (fewer test errors), as shown in Figure 4.

4.4 Domain Adaptation Results

Though MSR achieves impressive performance in CER reduction over the comparison methods, as described in Section 4.2, the experiments are all performed using newspaper text for both training and testing, which is not a realistic scenario if we are to deploy the model in an application. This section reports the results of additional experiments in which we adapt a model trained on one domain to a different domain, i.e., in a so-called cross-domain LM adaptation paradigm. See (Suzuki and Gao 2005) for a detailed report.

The data sets we used stem from five distinct sources of text. The Nikkei newspaper corpus described in Section 4.1 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). For each of the four domains, we used an 72,000-sentence subset as adaptation training data, a 5,000-sentence subset as held-out data and another 5,000-sentence subset as test data. Similarly, all corpora have been word-segmented, and we kept for each training sample, in the four adaptation domains, the best 20 hypotheses in its candidate conversion set for discriminative training.

We compared MSR with three other LM adaptation methods:

Baseline is the background word trigram model, as described in Section 4.1.

MAP (maximum a *posteriori*) 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 $P(w_i|h) = \lambda P_b(w_i|h) + (1-\lambda)P_a(w_i|h)$, 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.

Perceptron, **Boosting** and **MSR** are the three discriminative methods described in the previous sections. For each of them, the base feature was

Model	Yomiuri	TuneUp	Encarta	Shincho
Baseline	3.70	5.81	10.24	12.18
MAP	3.69	5.47	7.98	10.76
MSR	**2.73**	**5.15**	**7.40**	**10.16**
Boosting	2.78	5.33	7.53	10.25
Perceptron	2.78	5.20	7.44	10.18

Table 2. CER(%) results on four adaptation test sets .

derived from the word trigram model trained on the background data, and other n-gram features (i.e. f_d, d = 1…D in Equation (2)) were trained on adaptation data. That is, the parameters of the background model are adjusted in such a way that minimizes the errors on adaptation data made by background model.

Results are summarized in Table 2. First of all, in all four adaptation domains, discriminative methods outperform MAP significantly. Secondly, the improvement margins of discriminative methods over MAP correspond to the similarities between background domain and adaptation domains. When the two domains are very similar to the background domain (such as Yomiuri), discriminative methods outperform MAP by a large margin. However, the margin is smaller when the two domains are substantially different (such as Encarta and Shincho). 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. However, discriminative methods do not have this limitation for they aim to reduce errors directly. Finally, we find that in most adaptation test sets, MSR achieves slightly better CER results than the two competing discriminative methods. Specifically, the improvements of MSR are statistically significant over the boosting method in three out of four domains, and over the perceptron algorithm in the Yomiuri domain. The results demonstrate again that MSR is robust.

5 Related Work

Discriminative models have recently been proved to be more effective than generative models in some NLP tasks, e.g., parsing (Collins 2000), POS tagging (Collins 2002) and LM for speech recognition (Roark et al. 2004). In particular, the linear models, though simple and non-probabilistic in nature, are preferred to their probabilistic counterpart such as logistic regression. One of the reasons, as pointed out by Ng and Jordan (2002), is that the parameters of a discriminative model can be fit either to maximize the conditional likelihood on training data, or to minimize the training errors. Since the latter optimizes the objective function that the system is graded on, it is viewed as being more truly in the *spirit* of discriminative learning.

The MSR method shares the same motivation: to minimize the errors directly as much as possible. Because the error function on a finite data set is a step function, and cannot be optimized easily, previous research approximates the error function by loss functions that are suitable for optimization (e.g. Collins 2000; Freund et al. 1998; Juang et al. 1997; Duda et al. 2001). MSR uses an alternative approach. It is a simple heuristic training procedure to minimize training errors directly without applying any approximated loss function.

MSR shares many similarities with previous methods. The basic training algorithm described in Section 3.2 follows the general framework of multi-dimensional optimization (e.g., Press et al. 1992). The line search is an extension of that described in (Och 2003; Quirk et al. 2005. The extension lies in the way of handling large number of features and training samples. Previous algorithms were used to optimize linear models with less than 10 features. The feature selection method described in Section 3.4 is a particular implementation of the feature selection methods described in (e.g., Theodoridis and Koutroumbas 2003). The major difference between the MSR and other methods is that it estimates the effectiveness of each feature in terms of its expected training error reduction while previous methods used metrics that are loosely coupled with reducing training errors. The way of dealing with feature correlations in feature selection in Equation (7), was suggested by Finette et al. (1983).

6 Conclusion and Future Work

We show that MSR is a very successful discriminative training algorithm for LM. Our experiments suggest that it leads to significantly better conversion performance on the IME task than either the MLE method or the two widely applied discriminative methods, the boosting and perceptron methods. However, due to the lack of theoretical underpinnings, we are unable to prove that MSR will always succeed. This forms one area of our future work.

One of the most interesting properties of MSR is that it can optimize any objective function (whether its gradient is computable or not), such as error rate in IME or speech, BLEU score in MT, precision and recall in IR (Gao et al. 2005). In particular, MSR can be performed on large-scale training set with millions of candidate features. Thus, another area of our future work is to test MSR on wider varieties of NLP tasks such as parsing and tagging.

References

Collins, Michael. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with the perceptron algorithm. In *EMNLP 2002*.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *ICML 2000*.

Duda, Richard O, Hart, Peter E. and Stork, David G. 2001. *Pattern classification*. John Wiley & Sons, Inc.

Finette S., Blerer A., Swindel W. 1983. Breast tissue classification using diagnostic ultrasound and pattern recognition techniques: I. Methods of pattern recognition. *Ultrasonic Imaging*, Vol. 5, pp. 55-70.

Freund, Y, R. Iyer, R. E. Schapire, and Y. Singer. 1998. An efficient boosting algorithm for combining preferences. In *ICML'98*.

Gao, Jianfeng, Hisami Suzuki and Yang Wen. 2002. Exploiting headword dependency and predictive clustering for language modeling. In *EMNLP 2002*.

Gao, J, H. Qin, X. Xiao and J.-Y. Nie. 2005. Linear discriminative model for information retrieval. In *SIGIR*.

Jelinek, Fred. 1997. *Statistical methods for speech recognition.* MIT Press, Cambridge, Mass.

Juang, B.-H., W.Chou and C.-H. Lee. 1997. Minimum classification error rate methods for speech recognition. *IEEE Tran. Speech and Audio Processing* 5-3: 257-265.

Ng, A. N. and M. I. Jordan. 2002. On discriminative vs. generative classifiers: a comparison of logistic regression and naïve Bayes. In *NIPS 2002:* 841-848.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *ACL 2003*

Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. 1992. *Numerical Recipes In C: The Art of Scientific Computing*. New York: Cambridge Univ. Press.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. In *ACL 2005*: 271-279.

Roark, Brian, Murat Saraclar and Michael Collins. 2004. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *ICASSP 2004*.

Suzuki, Hisami and Jianfeng Gao. 2005. A comparative study on language model adaptation using new evaluation metrics. In *HLT/EMNLP 2005*.

Theodoridis, Sergios and Konstantinos Koutroumbas. 2003. *Pattern Recognition*. Elsevier.

Vapnik, V. N. 1999. *The nature of statistical learning theory.* Springer-Verlag, New York.

A Salience Driven Approach to Robust Input Interpretation in Multimodal Conversational Systems

Joyce Y. Chai **Shaolin Qu**
Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
{jchai@cse.msu.edu, qushaoli@cse.msu.edu}

Abstract

To improve the robustness in multimodal input interpretation, this paper presents a new salience driven approach. This approach is based on the observation that, during multimodal conversation, information from deictic gestures (e.g., point or circle) on a graphical display can signal a part of the physical world (i.e., representation of the domain and task) of the application which is salient during the communication. This salient part of the physical world will prime what users tend to communicate in speech and in turn can be used to constrain hypotheses for spoken language understanding, thus improving overall input interpretation. Our experimental results have indicated the potential of this approach in reducing word error rate and improving concept identification in multimodal conversation.

1 Introduction

Multimodal conversational systems promote more natural and effective human machine communication by allowing users to interact with systems through multiple modalities such as speech and gesture (Cohen et al., 1996; Johnston et al., 2002; Pieraccini et al., 2004). Despite recent advances, interpreting what users communicate to the system is still a significant challenge due to insufficient recognition (e.g., speech recognition) and understanding (e.g., language understanding) performance. Significant improvement in the robustness of multimodal interpretation is crucial if multimodal systems are to be effective and practical for real world applications.

Previous studies have shown that, in multimodal conversation, multiple modalities tend to complement each other (Cassell et al. 1994). Fusing two or more modalities can be an effective means of reducing recognition uncertainties, for example, through mutual disambiguation (Oviatt 1999). For semantically-rich modalities such as speech and pen-based gesture, mutual disambiguation usually happens at the fusion stage where partial semantic representations from individual modalities are disambiguated and combined into an overall interpretation (Johnston 1998, Chai et al., 2004a). One problem is that some critical but low probability information from individual modalities (e.g., recognized alternatives with low probabilities) may never reach the fusion stage. Therefore, this paper addresses how to use information from one modality (e.g., deictic gesture) to directly influence the semantic processing of another modality (e.g., spoken language understanding) even before the fusion stage.

In particular we present a new salience driven approach that uses gesture to influence spoken language understanding. This approach is based on the observation that, during multimodal conversation, information from deictic gestures (e.g., point or circle) on a graphical interface can signal a part of the physical world (i.e., representation of the domain and task) of the application which is salient during the communication. This salient part of the physical world will prime what users tend to communicate in speech and thus in turn can be used to constrain hypotheses for spoken language understanding. In particular, this approach incorporates a notion of salience from deictic gestures into language models for spoken language processing. Our experimental results indicate the potential of this approach in reducing word error rate and improving concept identification from spoken utterances.

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 217–224, Vancouver, October 2005. ©2005 Association for Computational Linguistics

In the following sections, we first introduce the current architecture for multimodal interpretation. Then we describe our salience driven approach and present empirical results.

2 Input Interpretation

Input interpretation is the identification of semantic meanings in user inputs. In multimodal conversation, user inputs can come from multiple channels (e.g., speech and gesture). Thus, most work on input interpretation is based on semantic fusion that includes individual recognizers and a sequential integration processes as shown in Figure 1. In this approach, a system first creates possible partial meaning representations from recognized hypotheses (e.g., N-best lists) independently of other modalities. For example, suppose a user says "what is the price of this painting" and at the same time points to a position on the screen. The partial meaning representations from the speech input and the gesture input are shown in (a-b) in Figure 1. The system uses the partial meaning representations to disambiguate each other and combines compatible partial representations together into an overall semantic representation as in Figure1(c).

In this architecture, the partial semantic representations from individual modalities are crucial for mutual disambiguation during multimodal fusion. The quality of partial semantic representations depends on how individual modalities are processed. For example, if the speech input is recognized as "what is the prize of this pant", then the partial representation from the speech input will not be created in the first place. Without a candidate partial representation, it is not likely for multimodal fusion to reach an overall meaning of the input given this late fusion architecture.

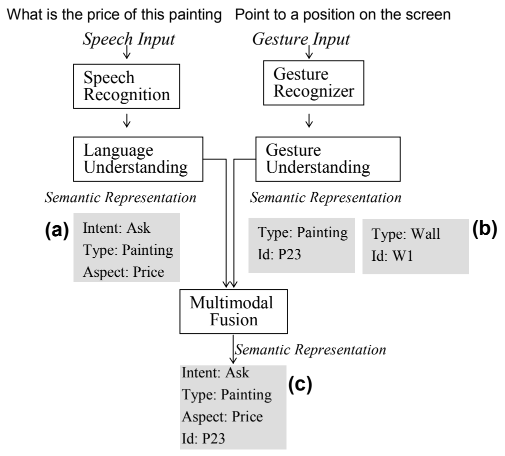


Figure 1: Semantics-based multimodal interpretation

Thus, a problem with the semantics-based fusion approach is that information from multiple modalities is only used during the fusion stage to disambiguate or combine partial semantic representations. This late use of information from other sources in the pipelined process can cause the loss of some low probability information (e.g., recognized alternatives with low probabilities which did not make it to the N-best list) which could be very crucial in terms of the overall interpretation. It is desirable to use information from multiple sources at an earlier stage before partial representations are created from individual modalities. For example, in ((Bangalore and Johnston 2000), a finite-state approach was applied to tightly couple multimodal language processing (e.g., gesture and speech) and speech recognition to improve recognition hypotheses. To further address this issue, in this paper, we present a salience driven approach that particularly applies gesture information (e.g., pen-based deictic gestures) to robust spoken language understanding before multimodal fusion.

3 Related Work on Salience Modeling

We first give a brief overview on the notion of salience and how salience modeling is applied in earlier work on natural language and multimodal language processing.

Linguistic salience describes the accessibility of entities in a speaker/hearer's memory and its implication in language production and interpretation. Many theories on linguistic salience have been developed, including how the salience of entities affects the form of referring expressions as in the Givenness Hierarchy (Gundel et al., 1993) and the local coherence of discourse as in the Centering Theory (Grosz et al., 1995). Salience modeling is used for both language generation and language interpretation; the latter is more relevant to our work. Most salience-based interpretation has focused on reference resolution for both linguistic referring expressions (e.g., pronouns) (Lappin and Leass 1995) and multimodal expressions (Hul et al. 1995; Eisenstein and Christoudias 2004).

Visual salience considers an object salient when it attracts a user's visual attention more than others. The cause of such attention depends on many factors including user intention, familiarity, and physical characteristics of objects. For example, an object may be salient when it has some properties the others do not have, such as it is the only one that is highlighted, or the only one of a certain size, category, or color

(Landragin et al., 2001). Visual salience can also be useful in input interpretation, for example, for multimodal reference resolution (Kehler 2000) and cross-modal coreference interpretation (Byron et al., 2005).

We believe that salience modeling should go beyond reference resolution. Our view is that the salience not only affects the use of referring expressions (and thus is useful for interpreting referring expressions), but also influences the linguistic context of the referring expressions. The spoken utterances that contain these expressions tend to describe information relating to the salient objects (e.g., properties or actions). Therefore, our goal in this paper is to take salience modeling a step further from reference resolution, towards overall language understanding.

4 A Salience Driven Approach

The new salience driven approach is based on the cognitive theory of Conversation Implicature (Grice 1975) and earlier empirical findings of user speech and gesture behavior in multimodal conversation (Oviatt 1999). The theory of Conversation Implicature (Grice 1975) states that speakers tend to make their contribution as informative as is required (for the current purpose of communication) and not make their contribution more informative than is required. In the context of multimodal conversation that involves speech and pen-based gesture, this theory indicates that users most likely will not make any unnecessary deictic gestures unless those gestures help in communicating users' intention. This is especially true since gestures usually take an extra effort from a user. When a pen-based gesture is intentionally delivered by a user, the information conveyed is often a crucial component in interpretation (Chai et al., 2005).

Speech and gesture also tend to complement each other. For example, when a speech utterance is accompanied by a deictic gesture (e.g., point or circle) on a graphical display, the speech input tends to issue commands or inquiries about properties of objects, and the deictic gestures tend to indicate the objects of interest. In addition, as shown in (Oviatt 1999), the deictic gestures often occur before spoken utterances. Our previous work (Chai et al., 2004b) also showed that 85% of time gestures occurred before corresponding speech units. Therefore, gestures can be used as an earlier indicator to anticipate the content of communication in the subsequent spoken utterances.

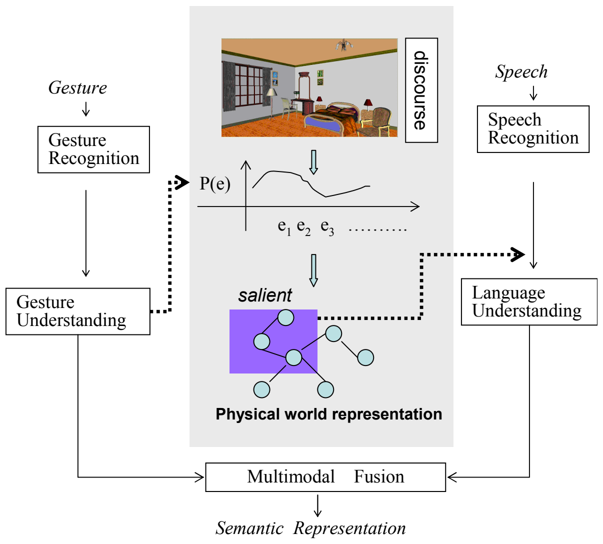


Figure 2: The salience driven approach: the salience distribution calculated from gesture is used to tailor language models for spoken language understanding

4.1 Overview

The general idea of the salience based approach is shown in Figure 2. For each application domain, there is a physical world representation that captures domain knowledge (details are described later). A deictic gesture can activate several objects on the graphical display. This activation will signal a distribution of objects that are salient. The salient objects are mapped to the physical world representation to indicate a salient part of representation that includes relevant properties or tasks related to the salient objects. This salient part of the physical world is likely to be the potential content of the spoken communication, and thus can be used to tailor language models for spoken language understanding. This process is shown in the middle shaded box of Figure 2. It bridges gesture understanding and language understanding at a stage before multimodal fusion. Note that the use of gesture information can be applied at different stages: during speech recognition to generate hypotheses or post processing of recognized hypotheses before language understanding. In this paper, we focus on the latter.

The physical world representation includes the following components:

• Domain Model. This component captures the relevant knowledge about the domain including domain objects, properties of the objects, relations between objects, and task models related to objects. Previous studies have shown that domain knowledge can be used to improve spoken language understanding (Wai et al, 2001). Currently, we apply a frame-based representation where a frame represents an object (or a type of object) in the domain and frame elements represent attributes and tasks related to the objects. Each frame element is associated with a semantic tag which indicates the semantic content of that element. In the future, the domain model might also include knowledge about the interface, for example, visual properties and spatial relations between objects on the interface.

• Domain Grammar. This component specifies grammar and vocabularies used to process language inputs. There are two types of representation. The first type is a semantics-based context free grammar where each non-terminal symbol represents a semantic tag (indicating semantic information such as the semantic type of an object, etc). Each word (i.e., the terminal symbol) in the lexicon relates to one or more semantic tags. Some of these semantic tags are directly linked to the frame elements in the domain model since they represent certain properties or tasks. This grammar was manually developed.

The second type of representation is based on annotated user spoken utterances. The data are annotated in terms of relevant semantic information (i.e., using semantic tags) in the utterance and the intended objects of interest (which are directly linked to the domain model). Based on the annotated data, N-grams can be learned to represent the dependency of language in our domain.

Based on the physical world representation, our approach supports the following operations:

Salience modeling. This operation calculates a salience distribution of entities in the physical world. In our current investigation, we limit the scope of entities to a closed set of objects from our physical world representation since the system has knowledge about those objects. These entities could have different salience values depending on whether they are visible on the graphical display, gestured by a user, or mentioned in the prior conversation. In this paper, we focus on the salience modeling using gesture information only.

Salience driven language understanding. This operation maps the salience distribution to the physical world representation and uses the salient world to influence spoken language understanding. Note that, in this paper, we are not concerned with acoustic models for speech recognition, but rather we are interested in the use of the salience distribution to prime language models and facilitate language understanding.

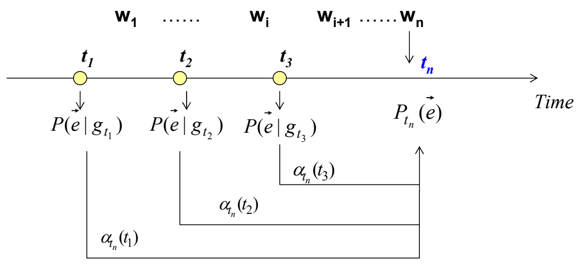


Figure 3: Salience modeling: the salience distribution at time t_n is calculated by a joint effect of gestures that happen before t_n.

4.2 Salience Modeling

We use a vector $\vec{e}$ to represent entities in the physical world representation. For each entity $e_k \in \vec{e}$, we use $P_{t_n}(e_k)$ to represent its salience value at time t_n. For all the entities, we use $P_{t_n}(\vec{e})$ to represent a salience distribution at time t_n. Figure 3 shows a sequence of words with corresponding gestures that occur at t_1, t_2, and t_3. As shown in Figure 3, the salience distribution at any given time t_n is influenced by a joint effect from this sequence of gestures that happen before t_n etc. Depending on its time of occurrence, each gesture may have a different impact on the salience distribution at time t_n. Note that although each gesture may have a short duration, here we only consider the beginning time of a gesture. Therefore, for an entity e_k, its salience value at time t_n is computed as follows:

$$P_{t_n}(e_k) = \frac{\sum_{i=1}^{m} \alpha_{t_n}(g_{t_i}) P(e_k | g_{t_i})}{\sum_{e \in \vec{e}} \sum_{i=1}^{m} \alpha_{t_n}(g_{t_i}) P(e | g_{t_i})} \qquad (1)$$

In Equation (1), m ($m \geq 1$) is the number of gestures that have occurred before t_n. The different impact of a gesture g_{t_i} at time t_i that contributes to the salience distribution at time t_n is represented as the weight $\alpha_{t_n}(g_{t_i})$ in Equation (1). Currently, we calculate the weight depending on the temporal distance as follows:

$$\alpha_{t_n}(g_{t_i}) = \exp[\frac{-(t_n - t_i)}{2000}] \quad (t_n \geq t_i) \qquad (2)$$

Equation (2) indicates that at a given time t_n (measured in milliseconds), the closer a gesture (at t_i) is to the time t_n, the higher impact this gesture has on the salience distribution (Chai et al., 2004b).

It is worth mentioning that a deictic gesture on the graphic display (e.g., pointing and circling) could have ambiguous interpretation by itself. For example, given an interface, a point or a circle on the screen could result in selection of different entities with different probabilities. Therefore, in Equation (1), $P(e \mid g_{t_i})$ is the selection probability which indicates the likelihood of selecting an entity *e* given a gesture at time t_i. This selection probability is calculated by a function of the distance between the location of the entity and the focus point of the recognized gesture on the display (Chai et al., 2004a). A normalization factor is incorporated to ensure that the summation of selection probabilities over all possible entities adds up to one.

When no gesture is involved in a given input, the salience distribution at any given time is a uniform distribution. If one or more gestures are involved, then Equation (1) is used to calculate the salience distribution.

4.3 Salience Driven Spoken Language Understanding

The salience distribution of entities identified based on the gesture information (as described above) is used to constrain hypotheses for language understanding. More specifically, for each onset of a spoken word at time *t* (i.e., the beginning time stamp of a spoken word), the salience distribution at *t* can be calculated based on a sequence of gestures that happen before *t* by Equation (1). This salience distribution can then be used to prime language models for spoken language processing.

Language Modeling

We first give a brief background of language modeling. Given an observed speech utterance O, the goal of speech recognition is to find a sequence of words W* so that $W^* = \arg\max P(O \mid W)P(W)$, where *P(O|W)* is the acoustic model and *P(W)* is the language model. In traditional speech recognition systems, the acoustic model provides the probability of observing the acoustic features given hypothesized word sequences and the language model provides the probability of a sequence of words. The language model is computed as follows:

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1 w_2)...P(w_n \mid w_1^{n-1})$$

Using the Markov assumption, the language model can be approximated by a bigram model as in:

$$P(w_1^n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

To improve the speech understanding results for spoken language interfaces, many systems have applied a loosely-integrated approach which decouples the language model from the acoustic model (Zue et al., 1991, Harper et al., 2000). This allows the development of powerful language models independent of the acoustic model, for example, utilizing topics of the utterances (Gildea and Hofmann 1999), syntactic or semantic labels (Heeman 1999), and linguistic structures (Chelba and Jelinek 2000, Wang and Harper 2002). Recently, we have seen work on language understanding based on environment (Schuler 2003) and language modeling using visual context (Roy and Mukherjee 2005). Our salience driven approach is inspired by this earlier work. Here, we do not address the acoustic model of speech recognition, but rather incorporate the salience distribution for language modeling. In particular, our focus is on investigating the effect of incorporating additional information from other modalities (e.g., gesture) with traditional language models.

Primed Language Model

The calculated salience distribution is used to prime the language model. More specifically, we use a class-based bigram model from (Brown et al, 1992):

$$P(w_i \mid w_{i-1}) = P(w_i \mid c_i)P(c_i \mid c_{i-1}) \quad (3)$$

In Equation (3), c_i is the class of the word w_i, which could be a syntactic class or a semantic class. $P(c_i \mid c_{i-1})$ is the class transition probability, which reflects the grammatical formation of utterances. $P(w_i \mid c_i)$ is the word class probability which measures the probability of seeing a word w_i given a class c_i. The class-based N-gram model can make better use of limited training data by clustering words into classes. A number of researchers have shown that the class-based N-gram model can successfully improve the performance of speech recognition (Jelinek 1990, Heeman 1999, Kneser and Ney 1993, Samuelsson and Reichl, 1999).

In our approach, the "class" used in the class-based bigram model comes from combined semantic and functional classes designed for our domain. For example, "this" is tagged as Demonstrative, and "price" is tagged as AttrPrice. As shown in Equation (3), there are two types of parameter estimation. In terms of the class transition probability, as in earlier work, we directly use the annotated data. In terms of the word class distribution, we incorporate the notion of salience. We use the salience distribution to dynamically adjust the world class probability $P(w_i \mid c_i)$ as follows:

$$P(w_i \mid c_i) = \sum_{e_k \in \vec{e}} \frac{P(w_i, c_i \mid e_k)}{P(c_i \mid e_k)} P_{t_i}(e_k) \qquad (4)$$

In Equation (4), $P_{t_i}(e_k)$ is the salience value for an entity e_k at time t_i (the onset of the spoken word w_i), which can be calculated by Equation (1). Equation (4) indicates that only information associated with the salient entities is used to estimate the word class distribution. In other words, the word class probability favors the salient physical world as indicated by the salience distribution $P_{t_i}(\vec{e})$. More specifically, at time t_i, given a semantic class c_i, the choice of word "w_i" is dependent on the salient physical world at the moment, which is represented as the salience distribution $P_{t_i}(\vec{e})$ at time t_i. For all w_i, the summation of this word class probability is equal to one. Furthermore, given an entity e_k, $P(w_i, c_i \mid e_k)$ and $P(c_i \mid e_k)$ are not dependent on time t_i, but rather on the domain and the use of language expressions. Therefore they can be estimated based on the training data that are annotated in terms of semantic information and the intended objects of interest (as discussed in Section 4.1). Since the annotated data is very limited, the sparse data can become a problem for the maximum likelihood estimation. Therefore, a smoothing technique based on the Katz backoff model (Katz, 1987) is applied. For example, to calculate $P(w_i, c_i \mid e_k)$, if a word w_i has one or more occurrences in the training data associated with the class c_i and the entity e_k, then its count is discounted by a fraction in the maximum likelihood estimation. If w_i does not occur, then this approach backs off to the domain grammar and redistributes the remaining probability mass uniformly among words in the lexicon that are linked with class c_i and entity e_k.

5 Evaluation

We evaluated the salience model during post processing recognized hypotheses. Given possible hypotheses from a speech recognizer, we use the salience-based language model to identify the most likely sequence of words. The salience distribution based on gesture was used to favor words that are consistent with the attention indicated by gestures.

The data collected from our previous user studies were used in our evaluation (Chai et al., 2004b). In these studies, users interacted with our multimodal interface using both speech and deictic gestures to find information about real estate properties. In particular, each user was asked to accomplish five tasks. Each of these tasks required the user to retrieve different types of information from our interface. For example, one task was to find the least expensive house in the most populated town. The data were recorded from eleven subjects including five non-native speakers and six native speakers. Each user's voice was individually trained before the study. Table 1 shows the relevant information about the data such as the total number of inputs (or turns) from each subject, the number of speech alone inputs without any gesture, and the baseline recognition results without using salience-based post processing in terms of the word error rate (WER). In total, we have collected 226 user inputs with an average of eight words per spoken utterance[1]. As shown in Table 1, the majority of inputs consisted of both speech and gesture. Since currently we only use gesture

User index	# of Inputs	# inputs w/o gesture	Baseline WER
1	21	0	0.287
2	31	0	0.335
3	27	0	0.399
4	10	0	0.680
5	8	1	0.200
6	36	0	0.387
7	18	0	0.250
8	25	1	0.278
9	23	0	0.482
10	11	0	0.117
11	16	3	0.255

Table 1: Related information about the evaluation data: user type, the number of turns per user, and the baseline word recognition rate.

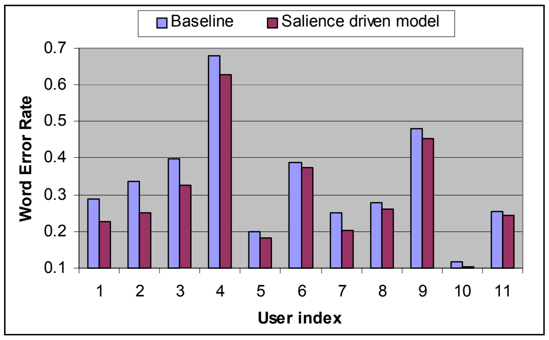


Figure 5: Comparison of the baseline and the result from post-processing in terms of WER

[1] The difference between the number of user inputs reported here and that in (Chai et al., 2004b) was caused by the situation where one intended user input (which was the unit for counting in our previous work) was split into a couple turns (which constitute the new counts here).

information in salience modeling, our approach will not affect speech only inputs.

To train the salience-based model, we applied the leave-one-out approach. The data from each user was held out as the testing data and the remaining users were used as the training data to acquire relevant probability estimations in Equation (3) and (4).

Figure 5 shows the comparison results between the baseline and the salience-based model in terms of word error rate (WER). The word error rate as a result of salience-based post processing is significantly better than that from the baseline recognizer (t = 4.75, p < 0.001). The average WER reduction is about 12%.

We further evaluated how the salience based model affects the final understanding results. This is because an improvement in WER may not directly lead to an improvement in understanding. We applied our semantic grammar on a sequence of words resulting from both the baseline and the salience-based post-processing to identify key concepts. In total, there were 686 concepts from the transcribed speech utterances. Table 2 shows the evaluation results. Precision measures the percentage of correctly identified concepts out of the total number of concepts identified based on a sequence of words. Recall measures the percentage of correctly identified concepts out of the total number of intended concepts from user's utterance. F-measurement combines precision and recall together as follows:

$$F = \frac{(\beta^2 + 1) \times \text{Precision} \times \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}, \quad where\, \beta = 1.$$

Table 2 shows that, on average, the concept identification based on the word sequence resulting from the salience-based approach performs better than the baseline in terms of both precision and recall. Figure 6 provides two examples to show the difference between the baseline recognition and the salience-based post processing.

The evaluation reported here is only an initial step based on a limited domain. The small scale in the number of objects and the vocabulary size can only demonstrate the potential of the salience-based approach to a limited degree. To further understand the advantages and issues of this approach, we are currently working on a more complex domain with richer concepts and relations, as well as larger vocabularies.

It is worth mentioning that the goal of this work is to explore whether salience modeling based on other modalities (e.g., gesture) can be used to prime traditional language models to facilitate spoken language processing. The salience driven approach based on additional modalities can be combined with more sophisticated language modeling (e.g., better parameter estimation) in the future.

User #	Baseline	Salience-based
Precision	80.3%	84.6%
Recall	75.7%	83.8%
F-measure	77.9%	84.2%

Table2. Overall concept identification comparison between the baseline and the salience driven model.

Example 1:
Transcription: What is the population of this town
Baseline recognition: What is the <u>publisher</u> of this <u>time</u>
Salience-based processing: what is the **population** of this **town**

Example 2:
Transcription: How much is this gray house
Baseline recognition: How much is this <u>great</u> house
Salience-based processing: How much is this **gray** house

Figure 6: Examples of utterances with baseline recognition and improved recognition from the salience-based processing.

6 Conclusion and Future Work

This paper presents a new salience driven approach to robust input interpretation in multimodal conversational systems. This approach takes advantage of rich information from multiple modalities. Information from deictic gestures is used to identify a part of the physical world that is salient at a given point of communication. This salient part of the physical world is then used to prime language models for spoken language understanding. Our experimental results have shown the potential of this approach in reducing word error rate and improving concept identification from spoken utterances in our application. Although currently we have only investigated the use of gesture information in salience modeling, the salience driven approach can be extended to include other modalities (e.g., eye gaze) and information (e.g., conversation context). Our future work will specifically investigate how to combine information from multiple sources in salience modeling and how to apply the salience models in different early stages of processing.

Acknowledgement

This work was supported by a CAREER grant IIS-0347548 from the National Science Foundation. The authors would like to thank anonymous reviewers for their helpful comments and suggestions.

References

Bangalore, S. and Johnston, M. 2000. Integrating Multimodal Language Processing with Speech Recognition. In *Proceedings of ICSLP*.

Brown, P., Della Pietra, V. J., deSouza, P. V., Lai, J. C, and Mercer, R. L. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467-479.

Byron, D., Mampilly, T., Sharma, V., and Xu, T. 2005. Utilizing Visual Attention for Cross-Modal Coreference Interpretation. *Spring Lecture Notes in Computer Science: Proceedings of Context-05*, page 83-96.

Cassell, J., Stone, M., Douville, B., Prevost, S., Achorn, B., Steedman, M., Badler, N., and Pelachaud, C. 1994. Modeling the interaction between speech and gesture. *Cognitive Science Society.*

Chai, J. Y., Prasov, Z., Blaim, J., and Jin, R. 2005. Linguistic Theories in Efficient Multimodal Reference Resolution: an Empirical Investigation. *The 10th International Conference on Intelligent User Interfaces (IUI-05),* pp. 43-50, San Diego, CA.

Chai, J. Y., Hong, P., Zhou, M. X, and Prasov, Z. 2004b. Optimization in Multimodal Interpretation. *In Proceedings of ACL*, pp. 1-8, Barcelona, Spain.

Chai, J. Y., Hong, P., and Zhou, M. 2004a. A Probabilistic Approach to Reference Resolution in Multimodal User Interfaces. *Proceedings of 9th International Conference on Intelligent User Interfaces (IUI-04)*, pp. 70-77, Madeira, Portugal.

Chelba, C. and Jelinek, F. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J.; Smith, I., Chen, L., and Clow, J. 1996. Quickset: Multimodal Interaction for Distributed Applications. *Proceedings of ACM Multimedia*, 31– 40.

Eisenstein J. and Christoudias. C. 2004. A salience-based approach to gesture-speech alignment. In *Proceedings of HLT/NAACL'04*.

Gildea, D. and Hofmann, T. 1999. Topic-based language models using EM. In *Proceedings of Eurospeech*.

Griffin, Z. M. 2001. Gaze durations during speech reflect word selection and phonological encoding. *Cognition* 82, B1-B14.

Grosz, B. J., Joshi, A. K., and Weinstein, S. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2).

Grice, H. P. Logic and Conversation. 1975. In Cole, P., and Morgan, J., eds. Speech Acts. New York, New York: Academic Press. 41-58.

Gundel, J. K., Hedberg, N., and Zacharski, R. 1993. Cognitive Status and the Form of Referring Expressions in Discourse. *Language* 69(2):274-307.

Harper, M.., White, C., Wang, W., Johnson, M., and Helzerman, R. 2000. The Effectiveness of Corpus-Induced Dependency Grammars for Post-processing Speech. *Proceedings of the North American Association for Computational Linguistics,* 102-109.

Heeman. P. 1999. POS tags and decision trees for language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Process (EMNLP).*

Huls, C., Bos, E., and Classen, W. 1995. Automatic Referent Resolution of Deictic and Anaphoric Expressions. *Computational Linguistics*, 21(1):59-79.

Jelinek, F. 1990. Self-organized language modeling for speech recognition. In Waibel, A. and Lee, K. F. (Eds). *Readings in Speech Recognition*, pp. 450-506.

Johnston, M. 1998. Unification-based Multimodal parsing, *Proceedings of COLING-ACL.*

Johnston, M., Bangalore, S., Visireddy G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., and Maloor, P. 2002. MATCH: An Architecture for Multimodal Dialog Systems, in *Proceedings of the 40th ACL*, Philadelphia, pp. 376-383.

Katz, S. M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3).

Kehler, A. 2000. Cognitive Status and Form of Reference in Multimodal Human-Computer Interaction, *Proceedings of AAAI'01.*

Kneser, R. and Ney, H. 1993. Improved clustering techniques for class-based statistical language modeling. In *Eurospeech'93*, pp. 973-976.

Landragin, F., Bellalem, N., and Romary, L. 2001. Visual Salience and Perceptual Grouping in Multimodal Interactivity. In: *First International Workshop on Information Presentation and Natural Multimodal Dialogue*, Verona, Italy, pp. 151-155.

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

Oviatt, S. 1999. Mutual Disambiguation of Recognition Errors in a Multimodal Architecture. In *Proceedings of CHI.*

Pieraccini, R., Dayandhi, K., Bloom, J., Dahan, J.-G., Phillips, M., Goodman, B. R., Prasad, K. V., 2004. Multimodal Conversational Systems for Automobiles, *Communications of the ACM*, Vol. 47, No. 1, pp. 47-49

Roy, D. and Mukherjee, N. 2005. Towards Situated Speech Understanding: Visual Context Priming of Language Models. *Computer Speech and Language*, 19(2): 227-248.

Samuelsson, C. and Reichl, W. 1999. A class-based Language Model for Large Vocabulary Speech Recognition Extracted from Part-of-Speech Statistics. In *IEEE ICASSP'99.*

Schuler, W. 2003. Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. *In Proceedings of ACL*, Sapporo, Japan.

Wai, C., Pierraccinni, R., and Meng, H. 2001. A Dynamic Semantic Model for Rescoring Recognition Hypothesis. *Proceedings of the ICASSP.*

Wang, W. and Harper. M. 2002. The superARV language model: In Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of EMNLP*, 238–247.

Zue, V., Glass, J., Goodine, D., Leung, H., Phillips, M., Polifroni, J., and Seneff, S. 1991. Integration of Speech Recognition and Natural Language Processing in the MIT Voyager System. *Proceedings of the ICASSP.*

Error Handling in the RavenClaw Dialog Management Framework

Dan Bohus
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, 15213
dbohus@cs.cmu.edu

Alexander I. Rudnicky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, 15213
air@cs.cmu.edu

Abstract

We describe the error handling architect-ture underlying the RavenClaw dialog management framework. The architecture provides a robust basis for current and future research in error detection and recovery. Several objectives were pursued in its development: task-independence, ease-of-use, adaptability and scalability. We describe the key aspects of architectural design which confer these properties, and discuss the deployment of this architect-ture in a number of spoken dialog systems spanning several domains and interaction types. Finally, we outline current research projects supported by this architecture.

1 Introduction

Over the last decade, improvements in speech recognition and other component technologies have paved the way for the emergence of complex task-oriented spoken dialog systems. While traditionally the research community has focused on building information-access and command-and-control systems, recent efforts aim towards building more sophisticated language-enabled agents, such as personal assistants, interactive tutors, open-domain question answering systems, etc. At the other end of the complexity spectrum, simpler systems have already transitioned into day-to-day use and are becoming the norm in the phone-based customer-service industry.

Nevertheless, a number of problems remain in need of better solutions. One of the most important limitations in today's spoken language interfaces is their lack of robustness when faced with understanding errors. This problem appears across all domains and interaction types, and stems primarily from the inherent unreliability of the speech recognition process. The recognition difficulties are further exacerbated by the conditions under which these systems typically operate: spontaneous speech, large vocabularies and user populations, and large variability in input line quality. In these settings, average word-error-rates of 20-30% (and up to 50% for non-native speakers) are quite common.

Left unchecked, speech recognition errors can lead to two types of problems in a spoken dialog system: *misunderstandings* and *non-understandings*. In a *misunderstanding,* the system obtains an incorrect semantic interpretation of the user's turn. In the absence of robust mechanisms for assessing the reliability of the decoded inputs, the system will take the misunderstanding as fact and will act based on invalid information. In contrast, in a *non-understanding* the system fails to obtain an interpretation of the input altogether. Although no false information is incorporated in this case, the situation is not much better: without an appropriate set of recovery strategies and a mechanism for diagnosing the problem, the system's follow-up options are limited and uninformed. In general, unless mitigated by accurate error awareness and robust recovery mechanisms, speech recognition errors exert a strong negative impact on the quality and ultimately on the success of the interactions (Sanders et al, 2002).

Two pathways towards increased robustness can be easily envisioned. One is to improve the accuracy of the speech recognition process. The second is to create mechanisms for detecting and gracefully handling potential errors at the conversation level. Clearly, these two approaches do not stand in opposition and a combined effort would lead to the best results. The error handling architecture we describe in this paper embodies the second approach: it aims to provide the mechanisms for robust error handling at the dialog management level of a spoken dialog system.

The idea of handling errors through conversation has already received a large amount of attention from the research community. On the theoretical side, several models of grounding in communication have been proposed (Clark and Schaefer, 1989; Traum, 1998). While these models provide useful insights into the grounding process as it happens in human-human communication, they lack the decision-making aspects required to drive the interaction in a real-life spoken dialog system. In the Conversational Architectures project, Paek and Horvitz (2000) address this challenge by developing a computational implementation of the grounding process using Bayesian belief networks. However, questions still remain: the structure and parameters of the belief networks are handcrafted, as are the utilities for the various grounding actions; due to scalability and task-representation issues, it is not known yet how the proposed approach would transfer and scale to other domains.

Three ingredients are required for robust error handling: (1) the ability to detect the errors, (2) a set of error recovery strategies, and (3) a mechanism for engaging these strategies at the appropriate time. For some of these issues, various solutions have emerged in the community. For instance, systems generally rely on recognition confidence scores to detect potential misunderstandings (e.g. Krahmer et al., 1999; Walker et al., 2000) and use explicit and implicit confirmation strategies for recovery. The decision to engage these strategies is typically based on comparing the confidence score against manually preset thresholds (e.g. Kawahara and Komatani, 2000). For non-understandings, detection is less of a problem (systems know by definition when non-understandings occur). Strategies such as asking the user to repeat or rephrase, providing help, are usually engaged via simple heuristic rules.

At the same time, a number of issues remain unsolved: can we endow systems with better error awareness by integrating existing confidence annotation schemes with correction detection mechanisms? Can we diagnose the non-understanding errors on-line? What are the tradeoffs between the various non-understanding recovery strategies? Can we construct a richer set of such strategies? Can we build systems which automatically tune their error handling behaviors to the characteristics of the domains in which they operate?

We have recently engaged in a research program aimed at addressing such issues. More generally, our goal is to develop a task-independent, easy-to-use, adaptive and scalable approach for error handling in task-oriented spoken dialog systems. As a first step in this program, we have developed a modular error handling architecture, within the larger confines of the RavenClaw dialog management framework (Bohus and Rudnicky, 2003). The proposed architecture provides the infrastructure for our current and future research on error handling. In this paper we describe the proposed architecture and discuss the key aspects of architectural design which confer the desired properties. Subsequently, we discuss the deployment of this architecture in a number of spoken dialog systems which operate across different domains and interaction types, and we outline current research projects supported by the proposed architecture.

2 RavenClaw Dialog Management

We begin with a brief overview of the RavenClaw dialog management framework, as it provides the larger context for the error handling architecture.

RavenClaw is a dialog management framework for task-oriented spoken dialog systems. To date, it has been used to construct a large number of systems spanning multiple domains and interaction types (Bohus and Rudnicky, 2003): information access (RoomLine, the Let's Go Bus Information System), guidance through procedures (LARRI), command-and-control (TeamTalk), taskable agents (Vera). Together with these systems, RavenClaw provides the larger context as well as a test-bed for evaluating the proposed error handling architecture. More generally, RavenClaw provides a robust basis for research in various other aspects of dialog management, such as learning at the task and discourse levels, multi-participant dialog, timing and turn-taking, etc.

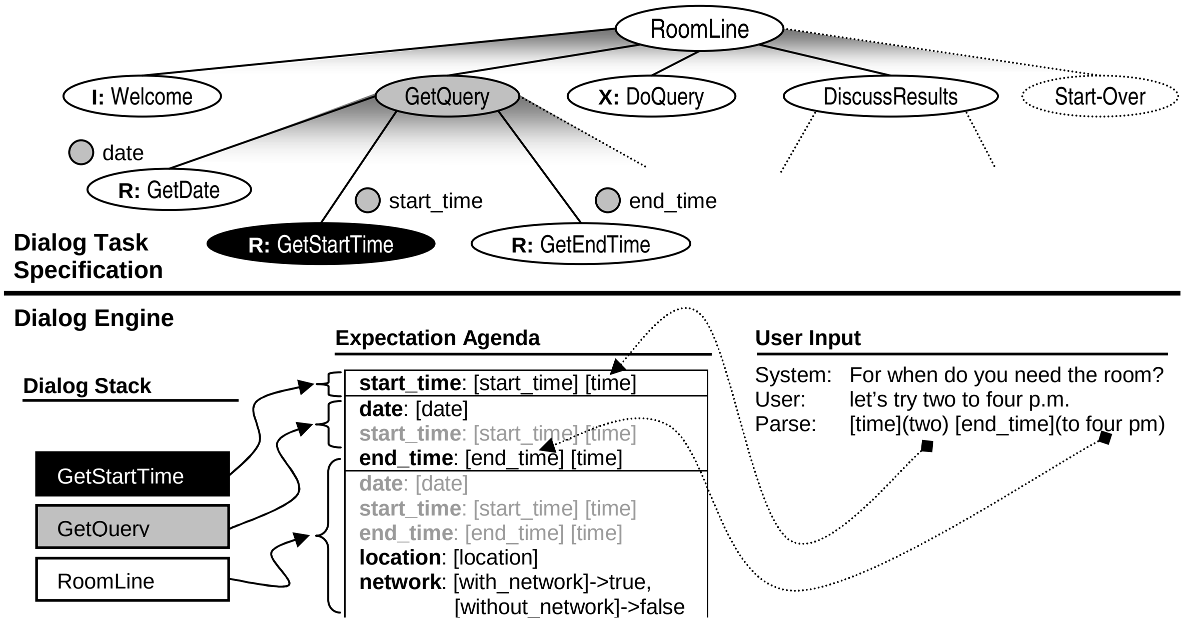
A key characteristic of the RavenClaw framework is the separation it enforces between the domain-specific and domain-independent aspects of dialog control. The domain-specific dialog control logic is described by a *Dialog Task Specification*, essentially a hierarchical dialog plan provided by the system author. A fixed, domain-independent *Dialog Engine* manages the conversation by executing the given Dialog Task Specification. In the process, the Dialog Engine also contributes a set of domain-independent conversational skills, such as error handling (discussed extensively in Section 4), timing and turn-taking, etc. The system authoring effort is therefore minimized and focused entirely on the domain-specific aspects of dialog control.

2.1 The Dialog Task Specification

A Dialog Task Specification consists of a tree of *dialog agents*, where each agent manages a subpart of the interaction. Figure 1 illustrates a portion of the dialog task specification from RoomLine, a spoken dialog system which can assist users in making conference room reservations. The root node subsumes several children: Welcome, which produces an introductory prompt, GetQuery which obtains the time and room constraints from the user, DoQuery which performs the database query, and DiscussResults which handles the follow-up negotiation dialog. Going one level deeper in the tree, GetQuery contains GetDate which requests the date for the reservation, GetStartTime and GetEndTime which request the times, and so on. This type of hierarchical task representation has a number of advantages: it scales up gracefully, it can be dynamically extended at runtime, and it implicitly captures a notion of context in dialog.

The agents located at the leaves of the tree are called *basic dialog agents*, and each of them implements an atomic dialog action (dialog move). There are four types of basic dialog agents: ***I****nform* – conveys information to the user (e.g. Welcome), ***R****equest* – asks a question and expects an answer (e.g. GetDate), ***E****xpect* – expects information without explicitly asking for it, and ***EX****ecute* – implements a domain specific operation (e.g. DoQuery). The agents located at non-terminal positions in the tree are called *dialog agencies* (e.g. RoomLine, GetQuery). Their role is to plan for and control the execution of their sub-agents. For each agent in the tree, the system author may specify preconditions, completion criteria, effects and triggers; various other functional aspects of the dialog agents (e.g. state-specific language models for request-agents, help-prompts) are controlled through parameters.

The information the system acquires and manipulates in conversation is captured in *concepts*, associated with various agents in the tree (e.g. date, start_time). Each concept maintains a history of previous values, information about current candidate hypotheses and their associated confidence scores, information about when the concept was last updated, as well as an extended set of flags which describe whether or not the concept has been conveyed to the user, whether or not the concept has been grounded, etc. This rich representation provides the necessary support for concept-level error handling.



RoomLine
I: Welcome
GetQuery
X: DoQuery
DiscussResults
Start-Over
date
R: GetDate
start_time
end_time
Dialog Task Specification
R: GetStartTime
R: GetEndTime
Dialog Engine
Expectation Agenda
User Input
Dialog Stack
GetStartTime
GetQuery
RoomLine
start_time: [start_time] [time]
date: [date]
start_time: [start_time] [time]
end_time: [end_time] [time]
date: [date]
start_time: [start_time] [time]
end_time: [end_time] [time]
location: [location]
network: [with_network]->true, [without_network]->false
System: For when do you need the room?
User: let's try two to four p.m.
Parse: [time](two) [end_time](to four pm)

Figure 1: RavenClaw architecture

2.2 The Dialog Engine

The Dialog Engine is the core domain-independent component which manages the interaction by executing a given Dialog Task Specification. The control algorithms are centered on two data-structures: a dialog stack, which captures the dialog structure at runtime, and an expectation agenda, which captures the system's expectations for the user input at each turn in the dialog. The dialog is controlled by interleaving *Execution Phases* with *Input Phases*.

During an Execution Phase, dialog agents from the tree are placed on, and executed from the dialog stack. At the beginning of the dialog, the root agent is placed on the stack. Subsequently, the engine repeatedly takes the agent on the top of the stack and executes it. When dialog agencies are executed, they typically schedule one of their subagents for execution by placing it on the stack. The dialog stack will therefore track the nested structure of the dialog at runtime. Ultimately, the execution of the basic dialog agents on the leaves of the tree generates the system's responses and actions.

During an Input Phase, the system assembles the expectation agenda, which captures what the system expects to hear from the user in a given turn. The agenda subsequently mediates the transfer of semantic information from the user's input into the various concepts in the task tree. For the interested reader, these mechanisms are described in more detail in (Bohus and Rudnicky, 2003)

Additionally, the Dialog Engine automatically provides a number of conversational strategies, such as the ability to handle various requests for help, repeating the last utterance, suspending and resuming the dialog, starting over, reestablishing the context, etc. These strategies are implemented as *library dialog agencies*. Their corresponding sub-trees are automatically added to the Dialog Task Specification provided by the system author (e.g. the Start-Over agency in Figure 1.) The automatic availability of these strategies lessens development efforts and ensures a certain uniformity of behavior both within and across tasks.

3 The Error Handling Architecture

The error handling architecture in the RavenClaw dialog management framework subsumes two main components: (1) a set of *error handling strategies* (e.g. explicit and implicit confirmation, asking the user to repeat, etc.) and (2) an *error handling process* which engages these strategies.

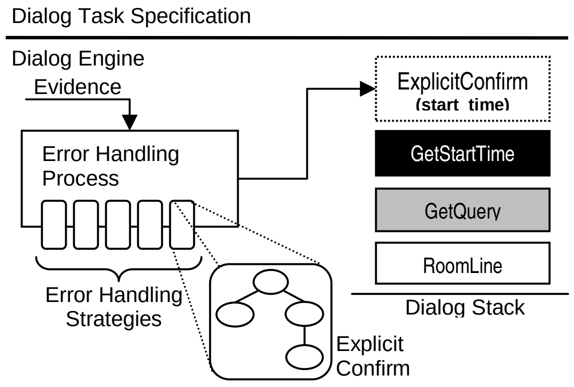


Figure 2: Error Handling – Block Diagram

The error handling strategies are implemented as library dialog agents. The decision process which engages these strategies is part of the Dialog Engine. This design, in which both the strategies and the decision process are decoupled from the dialog task, as well as from each other, provides a number of advantages. First, it ensures that the error handling mechanisms are reusable across different dialog systems. Second, the approach guarantees a certain uniformity and consistency in error handling behaviors both within and across systems. Third, as new error handling strategies are developed, they can be easily plugged into any existing system. Last, but not least, the approach significantly lessens the system authoring effort by allowing developers to focus exclusively on describing the dialog control logic.

The responsibility for handling potential understanding errors[1] is delegated to the Error Handling Process which runs in the Dialog Engine (see Figure 2). At each system turn, this process collects evidence and makes a decision with respect to engaging any of the error handling strategies. When necessary, it will insert an error handling strategy on the dialog stack (e.g. the ExplicitConfirm (start_time) strategy in Figure 2), thus modifying on-the-fly the task originally specified by the system author. The strategy executes and, once completed, it is removed from the stack and the dialog resumes from where it was left off.

[1] Note that the proposed framework aims to handle understanding errors. The corresponding strategies are generic and can be applied in any domain. Treatment of domain or task-specific errors (e.g. database access error, etc) still needs to be implemented as part of the dialog task specification.

3.1 Error Handling Strategies

The error handling strategies can be divided into two groups: strategies for handling potential misunderstandings and strategies for handling non-understandings.

For handling potential misunderstandings, three strategies are currently available: *Explicit Confirmation*, *Implicit Confirmation* and *Rejection*.

For non-understandings, a larger number of error recovery strategies are currently available: *AskRepeat* – the system asks the user to repeat; *AskRephrase* – the system asks the user to rephrase; *Reprompt* – the system repeats the previous prompt; *DetailedReprompt* – the system repeats a more verbose version of the previous prompt, *Notify* – the system simply notifies the user that a non-understanding has occurred; *Yield* – the system remains silent, and thus implicitly notifies the user that a non-understanding has occurred; *MoveOn* – the system tries to advance the task by giving up on the current question and moving on with an alternative dialog plan (note that this strategy is only available at certain points in the dialog); *YouCanSay* – the system gives an example of what the user could say at this point in the dialog; *FullHelp* – the system provides a longer help message which includes an explanation of the current state of the system, as well as what the user could say at this point. An in-depth analysis of these strategies and their relative tradeoffs is available in (Bohus and Rudnicky, 2005a). Several sample dialogs illustrating these strategies are available on-line (RoomLine, 2003).

3.2 Error Handling Process

The error handling decision process is implemented in a distributed fashion, as a collection of local decision processes. The Dialog Engine automatically associates a local error handling process with each concept, and with each request agent in the dialog task tree, as illustrated in Figure 3. The error handling processes running on individual concepts are in charge of recovering from misunderstandings on those concepts. The error handling processes running on individual request agents are in charge or recovering from non-understandings on the corresponding requests.

At every system turn, each concept- and request-agent error handling process computes and forwards its decision to a gating mechanism, which queues up the actions (if necessary) and executes them one at a time. For instance, in the example in Figure 3, the error handling decision process for the start_time concept decides to engage an explicit confirmation on that concept, while the other decision processes do not take any action. In this case the gating mechanism creates a new instance of an explicit confirmation agency, passes it the pointer to the concept to be confirmed (start_time), and places it on the dialog stack. On completion, the strategy updates the confidence score of the confirmed hypothesis in light of the user response, and the dialog resumes from where it was left off.

The specific implementation of the local decision processes constitutes an active research issue. Currently, they are modeled as Markov Decision Processes (MDP). The error handling processes running on individual concepts (*concept-MDPs* in

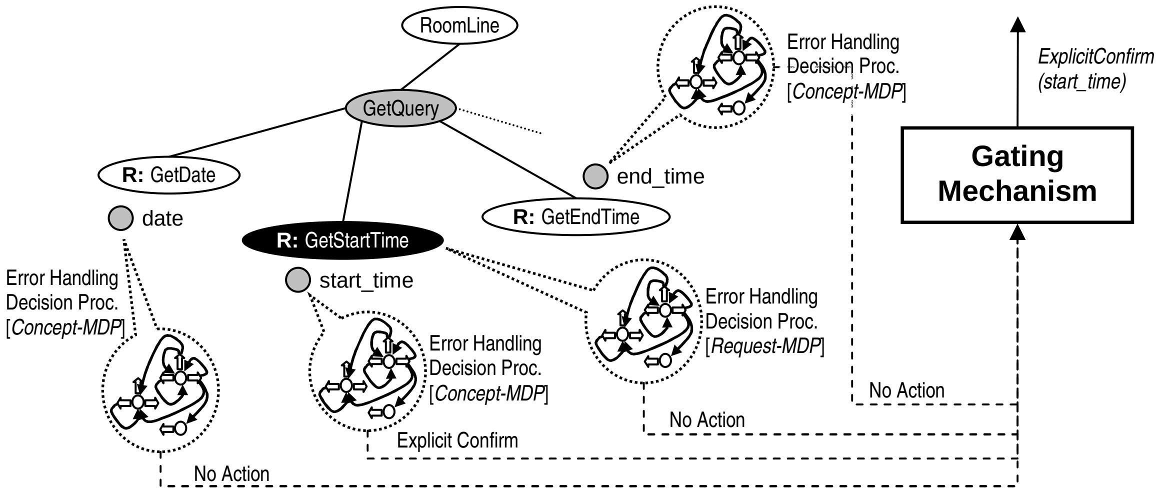

RoomLine
GetQuery
R: GetDate
date
R: GetStartTime
start_time
R: GetEndTime
end_time
Error Handling Decision Proc. [Concept-MDP]
Error Handling Decision Proc. [Concept-MDP]
Error Handling Decision Proc. [Request-MDP]
Error Handling Decision Proc. [Concept-MDP]
Gating Mechanism
ExplicitConfirm (start_time)
No Action
No Action
Explicit Confirm
No Action

Figure 3: A Distributed Error Handling Process

Figure 3) are partially-observable MDPs, with 3 underlying hidden states: c*orrect*, *incorrect* and *empty*. The belief state is constructed at each time step from the confidence score of the top-hypothesis for the concept. For instance, if the top hypothesis for the start_time concept is 10 a.m. with confidence 0.76, then the belief state for the POMDP corresponding to this concept is: {P(correct)=0.76, P(incorrect)=0.24, P(empty)=0}. The action-space for these models contains the three error recovery strategies for handling potential misunderstandings, and *no-action*. The third ingredient in the model is the *policy*. A policy defines which action the system should take in each state, and is indirectly described by specifying the utility of each strategy in each state. Currently, a number of predefined policies (e.g. always-explicit-confirm, pessimistic, and optimistic) are available in the framework. Alternatively, system authors can specify and use their own policies.

The error handling processes running on request agents (*request-MDPs* in Figure 3) are in charge of handling non-understandings on those requests. Currently, two types of models are available for this purpose. The simplest model has three states: *non-understanding, understanding* and *inactive*. A second model also includes information about the number of consecutive non-understandings that have already happened. In the future, we plan to identify more features which carry useful information about the likelihood of success of individual recovery strategies and use them to create more complex models. The action-space is defined by the set of non-understanding recovery strategies presented in the previous subsection, and *no-action*. Similar to the concept-MDPs, a number of default policies are available; alternatively, system authors can specify their own policy for engaging the strategies.

While the MDP implementation allows us to encode various expert-designed policies, our ultimate goal is to learn such policies from collected data using reinforcement learning. Reinforcement learning has been previously used to derive dialog control policies in systems operating with small tasks (Scheffler and Young, 2002; Singh et al, 2000). The approaches proposed to date suffer however from one important shortcoming, which has so far prevented their use in large, practical spoken dialog systems. The problem is lack of scalability: the size of the state space grows very fast with the size of the dialog task, and this renders the approach unfeasible in complex domains. A second important limitation of reinforcement learning techniques proposed to date is that the learned policies cannot be reused across tasks. For each new system, a new MDP has to be constructed, new data has to be collected, and a new training phase is necessary. This requires a significant amount of expertise and effort from the system author.

We believe that the error handling architecture we have described addresses these issues in several ways. The central idea behind the distributed nature of the approach is to keep the learning problem tractable by leveraging independence relationships between different parts of the dialog. First, the state and action-spaces can be maintained relatively small since we are only focusing on making error handling decisions (as opposed to other dialog control decisions). A more complex task translates into a larger number of MDP instantiations rather than a more complex model structure. Second, both the model structure and parameters (i.e. the transition probabilities) can be tied across models: for instance the MDP for grounding the start_time concept can be identical to the one for grounding the end_time concept; all models for grounding Yes/No concepts could be tied together, etc. Model tying has the potential to greatly improve scalability since data is polled together and the total number of model parameters to be learned grows sub-linearly with the size of the task. Third, since the individual MDPs are decoupled from the actual system task, the policies learned in a particular system can potentially be reused in other systems (e.g. we expect that grounding yes/no concepts functions similarly at different locations in the dialog, and across domains). Last but not least, the approach can easily accommodate dynamic task generation. In traditional reinforcement learning approaches the state and action-spaces of the underlying MDP are task-specific. The task therefore has to be fixed, known in advance: for instance the slots that the system queries the user about (in a slot-filling system) are fixed. In contrast, in the RavenClaw architecture, the dialog task tree (e.g. the dialog plan) can be dynamically expanded at runtime with new questions and concepts, and the corresponding request- and concept-MDPs are automatically created by the Dialog Engine.

4 Deployment and Current Research

While a quantitative evaluation of design characteristics such as task-independence, scalability, and ease-of-use is hard to perform, a first-order empirical evaluation of the proposed error handling architecture can be accomplished by using it in different systems and monitoring the system authoring process and the system's operation.

To date, the architecture has been successfully deployed in three different spoken dialog systems. A first system, RoomLine (2003), is a phone-based mixed-initiative system that assists users in making conference room reservations on campus. A second system, the Let's Go! Bus Information System (Raux et al, 2005), provides information about bus routes and schedules in the greater Pittsburgh area (the system is available to the larger public). Finally, Vera is a phone-based taskable agent that can be instructed to deliver messages to a third party, make wake-up calls, etc. Vera actually consists of two dialog systems, one which handles incoming requests (Vera In) and one which performs message delivery (Vera Out). In each of these systems, the authoring effort with respect to error handling consisted of: (1) specifying which models and policies should be used for the concepts and request-agents in the dialog task tree, and (2) writing the language generation prompts for explicit and implicit confirmations for each concept.

Even though the first two systems operate in similar domains (information access), they have very different user populations: students and faculty on campus in the first case versus the entire Pittsburgh community in the second case. As a result, the two systems were configured with different error handling strategies and policies (see Table 1). RoomLine uses explicit and implicit confirmations with an optimistic policy to handle potential misunderstandings. In contrast, the Let's Go Public Bus Information System always uses explicit confirmations, in an effort to increase robustness (at the expense of potentially longer dialogs). For non-understandings, RoomLine uses the full set of non-understanding recovery strategies presented in section 3.1. The Let's Go Bus Information system uses the *YouCanSay* and *FullHelp* strategies. Additionally a new *GoToAQuieterPlace* strategy was developed for this system (and is now available for use into any other RavenClaw-based system). This last strategy asks the user to move to a quieter place, and was prompted by the observation that a large number of users were calling the system from noisy environments.

While the first two systems were developed by authors who had good knowledge of the RavenClaw dialog management framework, the third system, Vera, was developed as part of a class project, by a team of six students who had no prior experience with RavenClaw. Modulo an initial lack of documentation, no major problems were encountered in configuring the system for automatic error handling. Overall, the proposed error handling architecture adapted easily and provided the desired functionality in each of these domains: while new strategies and recovery policies were developed for some of the systems, no structural changes were required in the error handling architecture.

	RoomLine	Let's Go Public	Vera In / Out
Domain	room reservations	bus route information	task-able agent
Initiative type	mixed	system	mixed / mixed
Task size: #agents ; #concepts	110 ; 25	57 ; 19	29 ; 4 / 31 ; 13
Strategies for misunderstandings	explicit and implicit	explicit	explicit and implicit / explicit only
Policy for misunderstandings	optimistic	always-explicit	optimistic / always-explicit
Strategies for non-understandings	all strategies (see Section 3.1)	go-to-quieter-place, you-can-say, help	all strategies / repeat prompt
Policy for non-understandings	choose-random	author-specified heuristic policy	choose-random / always-repeat-prompt
Sessions collected so far	1393	2836	72 / 131
Avg. task success rate	75%	52%	(unknown)
% Misunderstandings	17%	28%	(unknown)
% Non-understandings	13%	27%	(unknown)
% turns when strategies engage	41%	53%	36% / 44%

Table 1: Spoken dialog systems using the RavenClaw error handling architecture

5 Conclusion and Future Work

We have described the error handling architecture underlying the RavenClaw dialog management framework. Its design is modular: the error handling strategies as well as the mechanisms for engaging them are decoupled from the actual dialog task specification. This significantly lessens the development effort: system authors focus exclusively on the domain-specific dialog control logic, and the error handling behaviors are generated transparently by the error handling process running in the core dialog engine. Furthermore, we have argued that the distributed nature of the error handling process leads to good scalability properties and facilitates the reuse of policies within and across systems and domains.

The proposed architecture represents only the first (but an essential step) in our larger research program in error handling. Together with the systems described above, it sets the stage for a number of current and future planned investigations in error detection and recovery. For instance, we have recently conducted an extensive investigation of non-understanding errors and the ten recovery strategies currently available in the RavenClaw framework. The results of that study fall beyond the scope of this paper and are presented separately in (Bohus and Rudnicky, 2005a). In another project supported by this architecture, we have developed a model for updating system beliefs over concept values in light of initial recognition confidence scores and subsequent user responses to system actions. Initially, our confirmation strategies used simple heuristics to update the system's confidence score for a concept in light of the user response to the verification question. We have showed that a machine learning based approach which integrates confidence information with correction detection information can be used to construct significantly more accurate system beliefs (Bohus and Rudnicky, 2005b). Our next efforts will focus on using reinforcement learning to automatically derive the error recovery policies.

References

Bohus, D., Rudnicky, A., 2003 – *RavenClaw: Dialogue Management Using Hierarchical Task Decomposition and an Expectation Agenda*, in Proceedings of Eurospeech-2003, Geneva, Switzerland

Bohus, D., Rudnicky, A., 2005a – *Sorry, I didn't Catch That! An Investigation into Non-understandings and Recovery Strategies*, to appear in SIGDial-2005, Lisbon, Portugal

Bohus, D., Rudnicky, A., 2005b – *Constructing Accurate Beliefs in Spoken Dialog Systems,* submitted to ASRU-2005, Cancun, Mexico

Clark, H.H., Schaefer, E.F., 1989 – *Contributing to Discourse*, in Cognitive Science, vol 13, 1989.

Kawahara, T., Komatani, K., 2000 – *Flexible mixed-initiative dialogue management using concept-level confidence measures of speech recognizer output*, in Proc. of COLING, Saarbrucken, Germany, 2000.

Krahmer, E., Swerts, M., Theune, M., Weegels, M., 1999 - *Error Detection in Human-Machine Interaction*, Speaking. From Intention to Articulation, MIT Press, Cambridge, Massachusetts, 1999

Paek, T., Horvitz, E., 2000 – *Conversation as Action Under Uncertainty*, in Proceedings of the Sixteenth Conference on Uncertainty and Artificial Intelligence, Stanford, CA, June 2000.

Raux, A., Langner, B., Bohus, D., Black, A., Eskenazi, M., 2005 – *Let's Go Public! Taking a Spoken Dialog System to the Real World*, submitted to Interspeech-2005, Lisbon, Portugal

RoomLine web site, as of June 2005 – www.cs.cmu.edu/~dbohus/RoomLine

Sanders, G., Le, A., Garofolo, J., 2002 – *Effects of Word Error Rate in the DARPA Communicator Data During 2000 and 2001*, in Proceedings of ICSLP'02, Denver, Colorado, 2002.

Scheffler, K., Young, S., 2002 – *Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning*, in Proceedings of HLT-2002.

Singh, S., Litman, D., Kearns, M., Walker, M., 2000 – *Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System*, in Journal of Artificial Intelligence Research, vol. 16, pp 105-133, 2000.

Traum, D., 1998 – *On Clark and Schaefer's Contribution Model and its Applicability to Human-Computer Collaboration*, in Proceedings of the COOP'98, May 1998.

Walker, M., Wright, J., Langkilde, I., 2000 – *Using Natural Language Processing and Discourse Features to Identify Understanding Errors in a Spoken Dialogue System*, in Proc. of the 17'th International Conference of Machine Learning, pp 1111-1118.

Effective Use of Prosody in Parsing Conversational Speech

Jeremy G. Kahn[†] **Matthew Lease**[*]
Eugene Charniak[*] **Mark Johnson**[*] **Mari Ostendorf**[†]

University of Washington, SSLI[†] Brown University[*]
{jgk,mo}@ssli.ee.washington.edu {mlease,ec,mj}@cs.brown.edu

Abstract

We identify a set of prosodic cues for parsing conversational speech and show how such features can be effectively incorporated into a statistical parsing model. On the Switchboard corpus of conversational speech, the system achieves improved parse accuracy over a state-of-the-art system which uses only lexical and syntactic features. Since removal of edit regions is known to improve downstream parse accuracy, we explore alternatives for edit detection and show that PCFGs are not competitive with more specialized techniques.

1 Introduction

For more than a decade, the Penn Treebank's Wall Street Journal corpus has served as a benchmark for developing and evaluating statistical parsing techniques (Collins, 2000; Charniak and Johnson, 2005). While this common benchmark has served as a valuable shared task for focusing community effort, it has unfortunately led to the relative neglect of other genres, particularly speech. Parsed speech stands to benefit from practically every application envisioned for parsed text, including machine translation, information extraction, and language modeling. In contrast to text, however, speech (in particular, conversational speech) presents a distinct set of opportunities and challenges. While new obstacles arise from the presence of speech repairs, the possibility of word errors, and the absence of punctuation and sentence boundaries, speech also presents a tremendous opportunity to leverage multi-modal input, in the form of acoustic or even visual cues.

As a step in this direction, this paper identifies a set of useful prosodic features and describes how they can be effectively incorporated into a statistical parsing model, ignoring for now the problem of word errors. Evaluated on the Switchboard corpus of conversational telephone speech (Graff and Bird, 2000), our prosody-aware parser out-performs a state-of-the-art system that uses lexical and syntactic features only. While we are not the first to employ prosodic cues in a statistical parsing model, previous efforts (Gregory et al., 2004; Kahn et al., 2004) incorporated these features as word tokens and thereby suffered from the side-effect of displacing words in the n-gram models by the parser. To avoid this problem, we generate a set of candidate parses using an off-the-shelf, k-best parser, and use prosodic (and other) features to rescore the candidate parses.

Our system architecture combines earlier models proposed for parse reranking (Collins, 2000) and filtering out edit regions (Charniak and Johnson, 2001). Detecting and removing edits prior to parsing is motivated by the claim that probabilistic context-free grammars (PCFGs) perform poorly at detecting edit regions. We validate this claim empirically: two state-of-the-art PCFGs (Bikel, 2004; Charniak and Johnson, 2005) are both shown to perform significantly below a state-of-the-art edit detection system (Johnson et al., 2004).

2 Previous Work

As mentioned earlier, conversational speech presents a different set of challenges and opportunities than encountered in parsing text. This paper focuses on the challenges associated with disfluencies (Sec. 2.1) and the opportunity of leveraging acoustic-prosodic cues at the sub-sentence level (Sec. 2.2). Here, sentence segmentation is assumed to be known (though punctuation is not available);

$$\text{...while}\quad \underbrace{\text{I think,}}_{\text{Reparandum}}\quad + \quad \underbrace{\text{uh, I mean,}}_{\text{Editing phrase}}\ \underbrace{\text{I know}}_{\text{Repair}}\ \text{that...}$$

Figure 1: The structure of a typical repair, with "+" indicating the interruption point.

the impact of automatic segmentation is addressed in other work (Kahn et al., 2004).

2.1 Speech Repairs and Parsing

Spontaneous speech abounds with disfluencies such as partial words, filled pauses (e.g., "uh", "um"), conversational fillers (e.g., "you know"), and parenthetical asides. One type of disfluency that has proven particularly problematic for parsing is speech repairs: when a speaker amends what he is saying mid-sentence (see Figure 1). Following the analysis of (Shriberg, 1994), a speech repair can be understood as consisting of three parts: the *reparandum* (the material repaired), the *editing phrase* (that is typically either empty or consists of a filler), and the *repair*. The point between the reparandum and the editing phrase is referred to as the *interruption point* (IP), and it is the point that may be acoustically marked. We refer to the reparandum and editing phrase together as an *edit* or *edit region*. Speech repairs are difficult to model with HMM or PCFG models, because these models can induce only linear or tree-structured dependencies between words. The relationship between reparandum and repair is quite different: the repair is often a "rough copy" of the reparandum, using the same or very similar words in roughly the same order. A language model characterizing this dependency with hidden stack operations is proposed in (Heeman and Allen, 1999).

Several parsing models have been proposed which accord special treatment to speech repairs. Most prior work has focused on handling disfluencies and continued to rely on hand-annotated transcripts that include punctuation, case, and known sentence boundaries (Hindle, 1983; Core and Schubert, 1999; Charniak and Johnson, 2001; Engel et al., 2002).

Of particular mention is the analysis of the relationship between speech repairs and parsing accuracy presented by Charniak and Johnson (2001), as this directly influenced our work. They presented evidence that improved edit detection (i.e. detecting the reparandum and edit phrase) leads to better parsing accuracy, showing a relative reduction in F-score error of 14% (2% absolute) between oracle and automatic edit removal. Thus, this work adopts their edit detection preprocessing approach. They have subsequently presented an improved model for detecting edits (Johnson et al., 2004), and our results here complement their analysis of the edit detection and parsing relationship, particularly with respect to the limitations of PCFGs in edit detection.

2.2 Prosody and parsing

While spontaneous speech poses problems for parsing due to the presence of disfluencies and lack of punctuation, there is information in speech associated with prosodic cues that can be taken advantage of in parsing. Certainly, prosodic cues are useful for sentence segmentation (Liu *et al.*, 2004), and the quality of automatic segmentation can have a significant impact on parser performance (Kahn et al., 2004). There is also perceptual evidence that prosody provides cues to human listeners that aid in syntactic disambiguation (Price *et al.*, 1991), and the most important of these cues seems to be the prosodic phrases (perceived groupings of words) or the boundary events marking them. However, the utility of sentence-internal prosody in parsing conversational speech is not well established.

Most early work on integrating prosody in parsing was in the context of human-computer dialog systems, where parsers typically operated on isolated utterances. The primary use of prosody was to rule out candidate parses (Bear and Price, 1990; Batliner *et al.*, 1996). Since then, parsing has advanced considerably, and the use of statistical parsers makes the candidate pruning benefits of prosody less important. This raises the question of whether prosody is useful for improving parsing accuracy for conversational speech, apart from its use in sentence

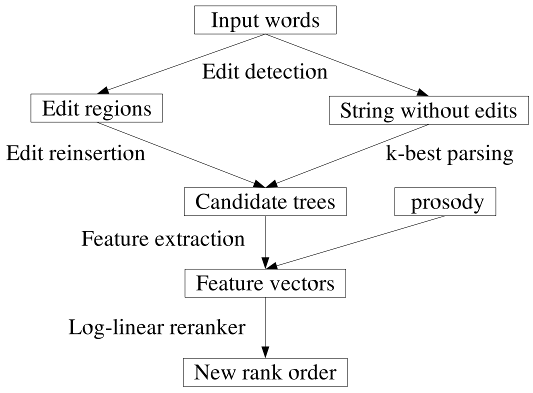


Figure 2: System architecture

boundary detection. Extensions of Charniak and Johnson (2001) look at using quantized combinations of prosodic features as additional "words", similar to the use of punctuation in parsing written text (Gregory et al., 2004), but do not find that the prosodic features are useful. It may be that with the short "sentences" in spontaneous speech, sentence-internal prosody is rarely of use in parsing. However, in edit detection using a parsing model (Johnson et al., 2004), posterior probabilities of automatically detected IPs based on prosodic cues (Liu *et al.*, 2004) are found to be useful. The seeming discrepancy between results could be explained if prosodic cues to IPs are useful but not other sub-sentence prosodic constituents. Alternatively, it could be that including a representation of prosodic features as terminals in (Gregory et al., 2004) displaces words in the parser n-gram model history. Here, prosodic event posteriors are used, with the goal of providing a more effective way of incorporating prosody than a word-like representation.

3 Approach

3.1 Overall architecture

Our architecture, shown in Figure 2, combines the parse reranking framework of (Collins, 2000) with the edit detection and parsing approach of (Charniak and Johnson, 2001). The system operates as follows:

1. Edit words are identified and removed.
2. Each resulting string is parsed to produce a set of k candidate parses.
3. Edit words reinserted into the candidates with a new part-of-speech tag EW. Consecutive sequences of edit words are inserted as single, flat EDITED constituents.
4. Features (syntactic and/or prosodic) are extracted for each candidate, i.e. candidates are converted to feature vector representation.
5. The candidates are rescored by the reranker to identify the best parse.

Use of Collins' parse reranking model has several advantages for our work. In addition to allowing us to incorporate prosody without blocking lexical dependencies, the discriminative model makes it relatively easy to experiment with a variety of prosodic features, something which is considerably more difficult to do directly with a typical PCFG parser.

Our use of the Charniak-Johnson approach of separately detecting disfluencies is motivated by their result that edit detection error degrades parser accuracy, but we also include experiments that omit this step (forcing the PCFG to model the edits) and confirm the practical benefit of separating responsibilities between the edit detection and parsing tasks.

3.2 Baseline system

We adopt an existing parser-reranker as our baseline (Charniak and Johnson, 2005). The parser component supports k-best parse generation, and the reranker component is used to rescore candidate parses proposed by the parser. In detail, the reranker selects from the set of k candidates $T = \{t_1, \ldots t_k\}$ the parse $t^\star \in T$ with the highest bracket F-score (in comparison with a hand-annotated reference). To accomplish this, a feature-extractor converts each candidate parse $t \in T$ into a vector of real-valued features $f(t) = (f_1(t), \ldots, f_m(t))$ (e.g., the value $f_j(t)$ of the feature f_j might be the number of times a certain syntactic structure appears in t). The reranker training procedure associates each feature f_j with a real-valued weight λ_j, and $\lambda' f(t)$ (the dot product of the feature vector and the weight vector λ) is a single scalar weight for each parse candidate. The reranker employs a maximum-entropy estimator that selects the λ that minimizes the log loss of the highest bracket F-score parse $t^\star$ conditioned on T (together with a Gaussian regularizer to prevent overtraining). Informally, λ is chosen to make high F-score parses as likely as possible under the (conditional) distribution defined by f and λ. As in (Collins, 2000), we generate training data for the reranker by reparsing the training corpus, using $n-1$ folds as training data to parse the n-th fold.

The existing system also includes a feature extractor that identifies interesting syntactic relationships not included in the PCFG parsing model (but used in the reranker). These features are primarily related to non-local dependencies, including parallelism of conjunctions, the number of terminals dominated by coordinated structures, right-branching root-to-leaf length, lexical/functional head pairs, n-gram style sibling relationships, etc.

3.3 Prosodic Features

Most theories of prosody have a symbolic representation for prosodic phrasing, where different combinations of acoustic cues (fundamental frequency, energy, timing) combine to give categorical perceptual differences. Our approach to integrating prosody in parsing is to use such symbolic boundary events, including prosodic break labels that build on linguistic notions of intonational phrases and hesitation phenomena. These events are predicted from a combination of continuous acoustic correlates, rather than using the acoustic features directly, because the intermediate representation simplifies training with high-level (sparse) structures. Just as phone-based acoustic models are useful in speech recognition systems as an intermediate level between words and acoustic features (especially for characterizing unseen words), the small set of prosodic boundary events are used here to simplify modeling the inter-dependent set of continuous-valued acoustic cues related to prosody. However, also as in speech recognition, we use posterior probabilities of these events as features rather than making hard decisions about presence vs. absence of a constituent boundary.

In the past, the idea of using perceptual categories has been dismissed as impractical due to the high cost of hand annotation. However, with advances in weakly supervised learning, it is possible to train prosodic event classifiers with only a small amount of hand-labeled data by leveraging information in syntactic parses of unlabeled data. Our strategy is similar to that proposed in (Nöth *et al.*, 2000), which uses categorical labels defined in terms of syntactic structure and pause duration. However, their system's category definitions are without reference to human perception, while we leverage learned relations between perceptual events and syntax with other acoustic cues, without predetermining the relation or requiring a direct coupling to syntax.

More specifically, we represent three classes of prosodic boundaries (or, breaks): major intonational phrase, hesitation, and all other word boundaries.[1] A small set of hand-labeled data from the treebanked portion of the Switchboard corpus (Ostendorf *et al.*, 2001) was used to train initial break prediction models based on both parse and acoustic cues. Next, the full set of treebanked Switchboard data is used with an EM algorithm that iterates between: i) finding probabilities of prosodic breaks in unlabeled data based on the current model, again using parse and acoustic features, and ii) re-estimating the model using the probabilities as weighted counts. Finally, a new acoustic-only break prediction model was designed from this larger data set for use in the parsing experiments.

In each stage, we use decision trees for models, in part because of an interest in analyzing the prosody-syntax relationships learned. The baseline system trained on hand-labeled data has error rates of 9.6% when all available cues are used (both syntax and prosody) and 16.7% when just acoustic and part-of-speech cues are provided (our target environment). Using weakly supervised (EM) training to incorporate unannotated data led to a 15% reduction in error rate to 14.2% for the target trees. The final decision tree was used to generate posteriors for each of the three classes, one for each word in a sentence.

¿From perceptual studies and decision tree analyses, we know that major prosodic breaks tend to co-occur with major clauses, and that hesitations often occur in edit regions or at high perplexity points in the word sequence. To represent the co-occurrence as a feature for use in parse reranking, we treat the prosodic break posteriors as weighted counts in accumulating the number of constituents in parse t of type i with break type j at their right edge, which (with some normalization and binning) becomes feature f_{ij}. Note that the unweighted count

[1]The intonational phrase corresponds to a break of "4" in the ToBI labeling system (Pitrelli et al., 1994), and a hesitation is marked with the "p" diacritic.

for constituent i corresponds directly to a feature in the baseline set, but the baseline set of features also includes semantic information via association with specific words. Here, we simply use syntactic constituents. It is also known that major prosodic breaks tend to be associated with longer syntactic constituents, so we used the weighted count strategy with length-related features as well. In all, the various attributes associated with prosodic break counts were the constituent label of the subtree, its length (in words), its height (maximal distance from the constituent root to any leaf), and the depth of the rightmost word (distance from the right word to the subtree root). For each type in each of these categories, there are three prosodic features, corresponding to the three break types.

3.4 Edit detection

To provide a competitive baseline for our parsing experiments, we used an off-the-shelf, state-of-the-art TAG-based model as our primary edit detector (Johnson et al., 2004).[2] This also provided us a competitive benchmark for contrasting the accuracy of PCFGs on the edit detection task (Section 4.2).

Whereas the crossing-dependencies inherent in speech repairs makes them difficult to model using HMM or PCFG approaches (Section 2.1), Tree Adjoining Grammars (TAGs) are capable of capturing these dependencies. To model both the crossed-dependencies of speech repairs and the linear or tree-structured dependencies of non-repaired speech, Johnson et al.'s system applies the noisy channel paradigm: a PCFG language model defines a probability distribution over non-repaired speech, and a TAG is used to model the optional insertion of edits. The output of this noisy channel model is a set of candidate edits which are then reranked using a max-ent model (similar to what is done here for parse reranking). This reranking step enables incorporation of features based on an earlier word-based classifier (Charniak and Johnson, 2001) in addition to output features of the TAG model. Acoustic features are not yet incorporated.

[2]We also evaluated another state-of-the-art edit detection system (Liu *et al.*, 2004) but found that it suffered from a mismatch between the current LDC specification of edits (LDC, 2004) and that used in the treebank.

4 Experimental design

4.1 Corpus

Experiments were carried out on conversational speech using the hand-annotated transcripts associated with the Switchboard treebank (Graff and Bird, 2000). As was done in (Kahn et al., 2004), we resegmented the treebank's sentences into V5-style sentence-like units (SUs) (LDC, 2004), since our ultimate goal was to be able to parse speech given automatically detected boundaries. Unfortunately, the original transcription effort did not provide punctuation guidelines, and the Switchboard treebanking was performed on the transcript unchanged, without reference to the audio. As a result, the sentence boundaries sometimes do not match human listener decisions using SU annotation guidelines, with differences mainly corresponding to treatment of discourse markers and backchannels. In the years since the original Switchboard annotation was performed, LDC has iteratively refined guidelines for annotating SUs, and significant progress has been made in automatically recovering SU boundaries annotated according to this standard (Liu *et al.*, 2004). To eventually leverage this work, we have taken the Meteer-annotated SUs (Meteer et al., 1995), for which there exists treebanked training data, and automatically adjusted them to be more like the V5 LDC standard, and resegmented the Switchboard treebank accordingly. In cases where the original syntactic constituents span multiple SUs, we discard any constituents violating the SU boundary, and in the event that an SU spans a treebank sentence boundary, a new top-level constituent SUGROUP is inserted to produce a proper tree (and evaluated like any other constituent in the gold tree).[3] While this SU resegmentation makes it difficult to compare our experimental results to past work, we believe this is a necessary step towards developing a more realistic baseline for fully automated parsing of speech.

In addition to resegmention, we removed all punctuation and case from the corpus to more closely reflect the form of output available from a speech recognizer. We retained partial words for consis-

[3]SU and treebank segments disagree at about 5% in each direction, due mostly to the analysis of discourse markers as conjunctions (sentences of >1 SU) and the separation of backchannels into separate treebank sentences (SUs of >1 sentence).

Table 1: Statistics on the Switchboard division used.

Section	Sides	SUs	Words
Train	1,031	87,599	659,437
Tune	126	13,147	103,500
Test	128	8,726	61,313
Total	1,285	109,472	824,250

tency with other work (Liu *et al.*, 2004; Johnson et al., 2004), although word fragments would not typically be available from ASR. Finally, of the 1300 total conversation sides, we discarded 15 for which we did not have prosodic data. Our corpus division statistics are given in Table 1. During development, experiments were carried out on the *tune* section; the *test* section was reserved for a final test run.

4.2 Experimental Variables

Our primary goal is to evaluate the extent to which prosodic cues could augment and/or stand-in for lexical and syntactic features. Correspondingly, we report on using three flavors of *feature extraction*: syntactic and lexical features (Section 3.2), prosodic features (Section 3.3), and both sets of features combined. For all three conditions, the first-stage score for each parse (generated by the off-the-shelf k-best parser) was always included as a feature.

A second parameter varied in the experiments was the method of upstream *edit detection* employed prior to parsing: PCFG, TAG-based, and oracle knowledge of treebank edit annotations. While it had been claimed that PCFGs perform poorly as edit detectors (Charniak and Johnson, 2001), we could not find empirical evidence in the literature quantifying the severity of the problem. Therefore, we evaluated two PCFGs (Bikel, 2004; Charniak and Johnson, 2005) on edit detection and compared their performance to a state-of-the-art TAG-based edit detection system (Johnson et al., 2004). For this experiment, we evaluated edit detection accuracy on a per-word basis, where any tree terminal is considered an edit word if and only if it is dominated by an EDITED constituent in the gold tree. The PCFGs were trained on the train section of the treebank data with the flattened edit regions included[4] and then used to parse the test data.[5] The TAG-based detector was trained on the same conversation sides, with its channel model trained on the Penn Treebank disfluency-annotated files and its language model trained on trees with the EDITED nodes excised. As shown in Table 2, we did find that both PCFGs performed significantly below the TAG-based detector.

[4]Training on flattened EDITED nodes improved detection accuracy for both PCFGs: as much as 15% for Bikel-Collins.

Table 2: Edit word detection performance for two word-based PCFGs and the TAG-based edit detector. F-score and error are word-based measures.

Edit Detector	Edit F-score	Edit Error
Bikel-Collins PCFG	65.3	62.1
Charniak PCFG	65.8	59.9
TAG-based	78.2	42.2

Table 3: Parsing F-score of various feature and edit-detector combinations.

	PCFG	TAG	Oracle
Edit F (Table 2)	65.8	78.2	100.0
Parser 1-best	84.4	85.0	86.9
Prosodic feats	85.0	85.6	87.6
Syntactic feats	85.9	86.4	88.4
Combined feats	86.0	86.6	88.6
Oracle-rate	92.6	93.2	95.2

5 Results

In evaluating parse accuracy, we adopt the *relaxed edited* revision (Charniak and Johnson, 2001) to the standard PARSEVAL metric, which penalizes systems that get EDITED spans wrong, but does not penalize disagreements in the attachment or internal structure of edit regions. This metric is based on the assumption that there is little reason to recover syntactic structure in regions of speech that have been repaired or restarted by the speaker.

Table 3 shows the F-scores for the top-ranked parses after reranking, where the first-stage PCFG parser was run with a beam-size of 50. The first and last rows show lower and upper bounds, respectively, for reranked parsing accuracy on each edit condition. As the oracle rate[6] shows, there is con-

[5]For the Charniak parser, edits were detected using only its PCFG component in 1-best mode, not its 2nd stage reranker.

[6]Oracle F uses the best parse in the 50-best list.

siderable room for improvement. Statistical significance was computed using a non-parametric shuffle test similar to that in (Bikel, 2004). For TAG and oracle *edit detection* conditions, the improvement from using the combined features over either prosodic or syntactic features in isolation was significant ($p < 0.005$). (For PCFG edit detection, $p < 0.04$.) Similarly, for all three *feature extraction* conditions, the improvement from using the TAG-based edit detector instead of the PCFG edit detector was also significant ($p < 0.001$). Interestingly, the TAG's 34% reduction in edit detection error relative to the PCFG yielded only about 23% of the parse accuracy differential between the PCFG and oracle conditions. Nevertheless, there remains a promising 2.0% difference in parse F-score between the TAG and oracle detection conditions to be realized by further improvements in edit detection. Training for the *syntactic* feature condition resulted in a learned weight λ with approximately 50K features, while the *prosodic* features used only about 1300 features. Despite this difference in the length of the λ vectors, the prosodic feature condition achieved 40–50% of the improvement of the syntactic features.

In some situations, e.g. for language modeling, improving the ordering and weights of the entire parse set (an not just the top ranked parse) is important. To illustrate the overall improvement of the reranked order, in Table 4 we report the reranked-oracle rate over the top s parses, varying the beam s. The error for each feature condition, relative to using the PCFG parser in isolation, is shown in Figure 3. Both the table and figure show that the reranked beam achieves a consistent trend in parse accuracy improvement relative to the PCFG beam, similar to what is demonstrated by the 1-best scores (Table 3).

Table 4: Reranked-oracle rate parse F-score for the top s parses with reference edit detection.

s	1	2	3	5	10	25
PCFG	86.9	89.8	91.0	92.2	93.4	94.6
Pros.	87.6	90.3	91.5	92.7	93.9	94.8
Syn.	88.4	91.3	92.4	93.4	94.3	95.0
Comb.	88.6	91.5	92.5	93.5	94.4	95.0

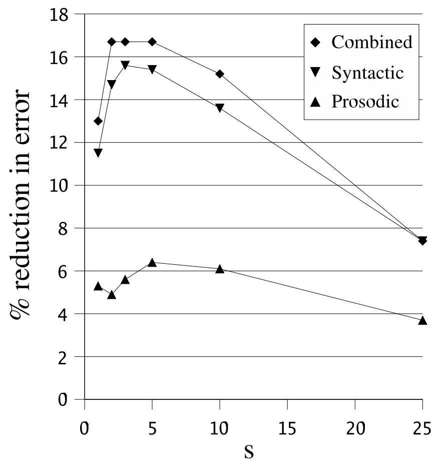


Figure 3: Reduction in error (Error $= 1 - F$) for the s-best reranked-oracle relative to the parser-only oracle, for different feature rerankings (reference edit detection).

6 Conclusion

This study shows that incorporating prosodic information into the parse selection process, along with non-local syntactic information, leads to improved parsing accuracy on accurate transcripts of conversational speech. Gains are shown to be robust to difficulties introduced by automatic edit detection and, in addition to improving the one-best performance, the overall ordering of the parse candidates is improved. While the gains from combining prosodic and syntactic features are not additive, since the prosodic features incorporates some constituent-structure information, the combined gains are significant. These results are consistent with related experiments with a different type of prosodically cued event, which showed that automatically detected IPs based on prosodic cues (Liu *et al.*, 2004) are useful in the reranking stage of a TAG-based speech repair detection system (Johnson et al., 2004).

The experiments described here used automatically extracted prosodic features in combination with human-produced transcripts. It is an open question as to whether the conclusions will hold for errorful ASR transcripts and automatically detected SU boundaries. However, there is reason to believe that relative gains from using prosody may be larger than those observed here for reference transcripts (though overall performance will degrade), based on prior work combining prosody and lexical cues to detect other language structures (Shriberg and Stolcke, 2004). While the prosody feature extraction depends on timing of the hypothesized word sequence, the acoustic cues are relatively robust to word errors and the break model can be retrained on recognizer output to automatically learn to discount the lexical evidence. Furthermore, if parse reranking operates on the top N ASR hypotheses, the reranking procedure can improve recognition outputs, as demonstrated in (Kahn, 2005) for syntactic features alone. Allowing for alternative SU hypotheses in reranking may also lead to improved SU segmentation.

In addition to assessing the impact of prosody in a fully automatic system, other avenues for future work include improving feature extraction. One could combine IP and prosodic break features (so far explored separately), find new combinations of prosody and syntactic structure, and/or incorporate other prosodic events. Finally, it may also be useful to integrate the prosodic events directly into the PCFG, in addition to their use in reranking.

This work was supported by the NSF under grants DMS-0074276, IIS-0085940, IIS-0112432, IIS-0326276, and LIS-9721276. Conclusions are those of the authors and do not necessarily reflect the views of the NSF.

References

A. Batliner *et al.* 1996. Prosody, empty categories and parsing - a success story. *Proc. ICSLP*, pp. 1169-1172.

J. Bear and P. Price. 1990. Prosody, syntax and parsing. *Proc. ACL*, pp. 17-22.

D. Bikel. 2004. *On the Parameter Space of Lexicalized Statistical Parsing Models*. Ph.D. thesis, U. Penn.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. *NAACL*, pp. 118-126.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proc. ACL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. *Proc. ICML*, pp. 175-182.

M. Core and L. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. *Proc. ACL*, pp. 413-420.

D. Engel, E. Charniak, and M. Johnson. 2002. Parsing and disfluency placement. *Proc. EMNLP*, pp. 49-54.

D. Graff and S. Bird. 2000. Many uses, many annotations for large speech corpora: Switchboard and TDT as case studies. *Proc. LREC*, pp. 427-433.

M. Gregory, M. Johnson, and E. Charniak. 2004. Sentence-internal prosody does not help parsing the way punctuation does. *Proc. NAACL*, pp. 81-88.

P. A. Heeman and J. F. Allen. 1999. Speech repairs, intonational phrases, and discourse markers: Modeling speaker's utterances in spoken dialogue. *Computational Linguistics*, 25(4):527-571.

D. Hindle. 1983. Deterministic parsing of syntactic nonfluencies. *Proc. ACL*, pp. 123-128.

M. Johnson, E. Charniak, and M. Lease. 2004. An improved model for recognizing disfluencies in conversational speech. *Proc. Rich Transcription Workshop*.

J. G. Kahn, M. Ostendorf, and C. Chelba. 2004. Parsing conversational speech using enhanced segmentation. *Proc. HLT-NAACL 2004*, pp. 125-128.

J. G. Kahn. 2005. *Moving beyond the lexical layer in parsing conversational speech.* M.A. thesis, U. Wash.

LDC. 2004. Simple metadata annotation specification. Tech. report, Linguistic Data Consortium. Available at http://www.ldc.upenn.edu/Projects/MDE.

Y. Liu *et al.* 2004. The ICSI-SRI-UW metadata extraction system. *Proc. ICSLP*, pp. 577-580.

M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer. 1995. Dysfluency annotation stylebook for the switchboard corpus. Tech. report, Linguistic Data Consortium.

E. Nöth *et al.* 2000. Verbmobil: The use of prosody in the linguistic components of a speech understanding system. *IEEE Trans. SAP*, 8(5):519-532.

M. Ostendorf *et al.* 2001. A prosodically labeled database of spontaneous speech. *ISCA Workshop on Prosody in Speech Recognition and Understanding*, pp. 119-121, 10.

J. Pitrelli, M. Beckman, and J. Hirschberg. 1994. Evaluation of prosodic transcription labeling reliability in the ToBI framework. *Proc. ICSLP*, pp. 123-126.

P. J. Price *et al.* 1991. The use of prosody in syntactic disambiguation. *JASA*, 90(6):2956-2970, 12.

E. Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, U.C. Berkeley.

E. Shriberg and A. Stolcke. 2004. Prosody modeling for automatic speech recognition and understanding. *Mathematical Foundations of Speech and Language Processing*. Springer-Verlag, pp. 105-114.

Automatically Learning Cognitive Status for Multi-Document Summarization of Newswire

Ani Nenkova and **Advaith Siddharthan** and **Kathleen McKeown**
Department of Computer Science
Columbia University
{ani,advaith,kathy}@cs.columbia.edu

Abstract

Machine summaries can be improved by using knowledge about the cognitive status of news article referents. In this paper, we present an approach to automatically acquiring distinctions in cognitive status using machine learning over the forms of referring expressions appearing in the input. We focus on modeling references to people, both because news often revolve around people and because existing natural language tools for named entity identification are reliable. We examine two specific distinctions—whether a person in the news can be assumed to be known to a target audience (hearer-old vs hearer-new) and whether a person is a major character in the news story. We report on machine learning experiments that show that these distinctions can be learned with high accuracy, and validate our approach using human subjects.

1 Introduction

Multi-document summarization has been an active area of research over the past decade (Mani and Maybury, 1999) and yet, barring a few exceptions (Daumé III et al., 2002; Radev and McKeown, 1998), most systems still use shallow features to produce an extractive summary, an age-old technique (Luhn, 1958) that has well-known problems. Extractive summaries contain phrases that the reader cannot understand out of context (Paice, 1990) and irrelevant phrases that happen to occur in a relevant sentence (Knight and Marcu, 2000; Barzilay, 2003). Referring expressions in extractive summaries illustrate this problem, as sentences compiled from different documents might contain too little, too much or repeated information about the referent.

Whether a referring expression is appropriate depends on the location of the referent in the hearer's mental model of the discourse—the referent's *cognitive status* (Gundel et al., 1993). If, for example, the referent is unknown to the reader at the point of mention in the discourse, the reference should include a description, while if the referent was known to the reader, no descriptive details are necessary.

Determining a referent's cognitive status, however, implies the need to model the intended audience of the summary. Can such a cognitive status model be inferred automatically for a general readership? In this paper, we address this question by performing a study with human subjects to confirm that reasonable agreement on the distinctions can be achieved between different humans (cf. §5). We present an automatic approach for inferring what the typical reader is likely to know about people in the news. Our approach uses machine learning, exploiting features based on the form of references to people in the input news articles (cf. §4). Learning cognitive status of referents is necessary if we want to ultimately generate new, more appropriate references for news summaries.

1.1 Cognitive status

In human communication, the wording used by speakers to refer to a discourse entity depends on their *communicative goal* and their beliefs about *what listeners already know*. The speaker's goals and beliefs about the listener's knowledge are both a part of a cognitive/mental model of the discourse.

Cognitive status distinctions depend on two parameters related to the referent—*a)* whether it already exists in the hearer's model of the discourse, and *b)* its degree of salience. The influence of these distinctions on the form of referring expressions has been investigated in the past. For example, centering theory (Grosz et al., 1995) deals predominantly with local salience (local attentional status), and the givenness hierarchy (information status) of Prince (1992) focuses on how a referent got in the discourse model (e.g. through a direct mention in the current discourse, through previous knowledge, or through inference), leading to distinctions such as discourse-old, discourse-new, hearer-old, hearer-new, inferable and containing inferable. Gundel et al. (1993) attempt to merge salience and givenness in a single hierarchy consisting of six distinctions in cognitive status (in focus, activated, familiar, uniquely identifiable, referential, type-identifiable).

Among the distinctions that have an impact on the form of references in a summary are the *familiarity* of the referent:

D. Discourse-old vs discourse-new

H. Hearer-old vs hearer-new

and its global salience[1]:

M. Major vs minor

In general, initial (discourse-new) references to entities are longer and more descriptive, while subsequent (discourse-old) references are shorter and have a purely referential function. Nenkova and McKeown (2003) have studied this distinction for references to people in summaries and how it can be used to automatically rewrite summaries to achieve better fluency and readability.

The other two cognitive status distinctions, whether an entity is central to the summary or not (major or minor) and whether the hearer can be assumed to be already familiar with the entity (hearer-old vs hearer-new status), have not been previously studied in the context of summarization. There is a tradeoff, particularly important for a short summary, between what the speaker wants to convey and how much the listener needs to know. The hearer-old/new distinction can be used to determine whether a description for a character is required from the listener's perspective. The major/minor distinction plays a role in defining the communicative goal, such as what the summary should be about and which characters are important enough to refer to by name.

[1]The notion of global salience is very important to summarization, both during content selection and during generation on initial references to entities. On the other hand, *in focus* or *local attentional state* are relevant to anaphoric usage during subsequent mentions.

1.2 Hearer-Old vs Hearer-New

Hearer-new entities in a summary should be described in necessary detail, while hearer-old entities do not require an introductory description. This distinction can have a significant impact on overall length and intelligibility of the produced summaries. Usually, summaries are very short, 100 or 200 words, for input articles totaling 5,000 words or more. Several people might be involved in a story, which means that if all participants are fully described, little space will be devoted to actual news. In addition, introducing already familiar entities might distract the reader from the main story (Grice, 1975). It is thus a good strategy to refer to an entity that can be assumed hearer-old by just a title + last name, e.g. *President Bush*, or by full name only, with no accompanying description, e.g. *Michael Jackson*.

1.3 Major vs Minor

Another distinction that human summarizers make is whether a character in a story is a major or a minor one and this distinction can be conveyed by using different forms of referring expressions. It is common to see in human summaries references such as *the dissident's father*. Usually, discourse-initial references solely by common noun, without the inclusion of the person's name, are employed when the person is not the main focus of a story (Sanford et al., 1988). By detecting the cognitive status of a character, we can decide whether to name the character in the summary. Furthermore, many summarization systems use the presence of named entities as a feature for computing the importance of a sentence (Saggion and Gaizaukas, 2004; Guo et al., 2003). The ability to identify the major story characters and use only them for sentence weighting can benefit such systems since only 5% of all people mentioned in the input are also mentioned in the summaries.

2 Why care about people in the news?

News reports (and consequently, news summaries) tend to have frequent references to people (in DUC data - see §3 for description - from 2003 and 2004, there were on average 3.85 references to people per 100-word human summary); hence it is important for news summarization systems to have a way of modeling the cognitive status of such referents and a theory for referring to people.

It is also important to note that there are differences in references to people between news reports and human summaries of news. Journalistic conventions for many mainstream newspapers dictate that initial mentions to people include a minimum description such as their role or title and affiliation. However, in human summaries, where there are greater space constraints, the nature of initial references changes. Siddharthan et al. (2004) observed that in DUC'04 and DUC'03 data[2], news reports contain on average one appositive phrase or relative clause every 3.9 sentences, while the human summaries contain only one per 8.9 sentences on average. In addition to this, we observe from the same data that the average length of a first reference to a named entity is 4.5 words in the news reports and only 3.6 words in human summaries. These statistics imply that human summarizers do compress references, and thus can save space in the summary for presenting information about the events. Cognitive status models can inform a system when such reference compression is appropriate.

3 Data preparation: the DUC corpus

The data we used to train classifiers for these two distinctions is the Document Understanding Conference collection (2001–2004) of 170 pairs of document input sets and the corresponding human-written multi-document summaries (2 or 4 per set). Our aim is to identify every person mentioned in the 10 news reports and the associated human summaries for each set, and assign labels for their cognitive status (hearer old/new and major/minor). To do this, we first preprocess the data (§3.1) and then perform the labeling (§3.2).

[2]The data provided under DUC for these years includes sets of about 10 news reports, 4 human summaries for each set, and the summaries by participating machine summarizers.

3.1 Automatic preprocessing

All documents and summaries were tagged with BBN's IDENTIFINDER (Bikel et al., 1999) for named entities, and with a part-of-speech tagger and simplex noun-phrase chunker (Grover et al., 2000). In addition, for each named entity, relative clauses, appositional phrases and copula constructs, as well as pronominal co-reference were also automatically annotated (Siddharthan, 2003). We thus obtained coreference information (cf. Figure 1) for each person in each set, across documents and summaries.

Andrei Sakharov	
Doc 1:	[IR] laureate Andrei D. Sakharov [CO] Sakharov [CO] Sakharov [CO] Sakharov [CO] Sakharov [PR] his [CO] Sakharov [PR] his [CO] Sakharov [RC] who acted as an unofficial Kremlin envoy to the troubled Transcaucasian region last month [PR] he [PR] He [CO] Sakharov
Doc 1:	[IR] Andrei Sakharov [AP] , 68 , a Nobel Peace Prize winner and a human rights activist , [CO] Sakharov [IS] a physicist [PR] his [CO] Sakharov

Figure 1: Example information collected for *Andrei Sakharov* from two news report. 'IR' stands for 'initial reference', 'CO' for noun co-reference, 'PR' for pronoun reference, 'AP' for apposition, 'RC' for relative clause and 'IS' for copula constructs.

The tools that we used were originally developed for processing single documents and we had to adapt them for use in a multi-document setting. The goal was to find, for each person mentioned in an input set, the list of all references to the person in both input documents and human summaries. For this purpose, all input documents were concatenated and processed with IDENTIFINDER. This was then automatically post-processed to mark-up coreferring names and to assign a unique canonical name (unique id) for each name coreference chain. For the coreference, a simple rule of matching the last name was used, and the canonical name was the "FirstName LastName" string where the two parts of the name could be identified [3]. Concatenating all documents assures that the same canonical name will be assigned to all named references to the same person.

[3]Occasionally, two or more different people with the same last name are discussed in the same set and this algorithm would lead to errors in such cases. We did keep a list of first names associated with the entity, so a more refined matching model could be developed, but this was not the focus of this work.

The tools for pronoun coreference and clause and apposition identification and attachment were run separately on each document. Then the last name of each of the canonical names derived from the IDENTIFINDER output was matched with the initial reference in the generic coreference list for the document with the last name. The tools that we used have been evaluated separately when used in normal single document setting. In our cross-document matching processes, we could incur more errors, for example when the general coreference chain is not accurate. On average, out of 27 unique people per cluster identified by IDENTIFINDER, 4 people and the information about them are lost in the matching step for a variety of reasons such as errors in the clause identifier, or the coreference.

3.2 Data labeling

Entities were automatically labeled as hearer-old or new by analyzing the syntactic form that human summarizers used for initial references to them. The labeling rests on the assumption that the people who produced the summaries used their own model of the reader when choosing appropriate references for the summary. The following instructions had been given to the human summarizers, who were not professional journalists: "To write this summary, assume you have been given a set of stories on a news topic and that your job is to summarize them for the general news sections of the Washington Post. Your audience is the educated adult American reader with varied interests and background in current and recent events." Thus, the human summarizers were given the freedom to use their assumptions about what entities would be generally hearer-old and they could refer to these entities using short forms such as (1) title or role+ last name or (2) full name only with no pre- or post-modification. Entities that the majority of human summarizers for the set referred to using form (1) or (2) were labeled as hearer-old. From the people mentioned in human summaries, we obtained 118 examples of hearer-old and 140 examples of hearer-new persons - 258 examples in total - for supervised machine learning.

In order to label an entity as major or minor, we again used the human summaries—entities that were mentioned *by name* in at least one summary were labeled *major*, while those not mentioned by name in any summary were labeled *minor*. The underlying assumption is that people who are not mentioned in any human summary, or are mentioned without being named, are not important. There were 258 major characters who made it to a human summary and 3926 minor ones that only appeared in the news reports. Such distribution between the two classes is intuitively plausible, since many people in news articles express opinions, make statements or are in some other way indirectly related to the story, while there are only a few main characters.

4 Machine learning experiments

The distinction between hearer-old and hearer-new entities depends on the readers. In other words, we are attempting to automatically infer which characters would be hearer-old *for the intended readership of the original reports*, which is also expected to be the intended readership of the summaries. For our experiments, we used the WEKA (Witten and Frank, 2005) machine learning toolkit and obtained the best results for hearer-old/new using a support vector machine (SMO algorithm) and for major/minor, a tree-based classifier (J48). We used WEKA's default settings for both algorithms.

We now discuss what features we used for our two classification tasks (cf. list of features in table 1). Our hypothesis is that features capturing the frequency and syntactic and lexical forms of references are sufficient to infer the desired cognitive model.

Intuitively, pronominalization indicates that an entity was particularly salient at a specific point of the discourse, as has been widely discussed in attentional status and centering literature (Grosz and Sidner, 1986; Gordon et al., 1993). Modified noun phrases (with apposition, relative clauses or premodification) can also signal different status.

In addition to the syntactic form features, we used two months worth of news articles collected over the web (and independent of the DUC collection we use in our experiments here) to collect unigram and bigram lexical models of first mentions of people. The names themselves were removed from the first mention noun phrase and the counts were collected over the premodifiers only. One of the lexical features we used is whether a person's description contains any of the 20 most frequent description words from our web corpus. We reasoned that these frequent de-

0,1:	Number of references to the person, including pronouns (total and normalized by feature 16)	2,3:	Number of times apposition was used to describe the person(total and normalized by feature 16)
4,5:	Number of times a relative clause was used to describe the person (total and normalized by 16)	6:	Number of times the entity was referred to by name after the first reference
7,8:	Number of copula constructions involving the person (total and normalized by feature 16)	9,10:	Number of apposition, relative clause or copula descriptions (total and normalized by feature 16)
11,12,13:	Probability of an initial reference according to the bigram model (av.,max and min of all initial references)	14:	Number of top 20 high frequency description words (from references to people in large news corpus) present in initial references
15:	Proportion of first references containing full name	16:	Total number of documents containing the person
17,18:	Number of appositives or relative clause attaching to initial references (total and normalized by feature 16)		

Table 1: List of Features provided to WEKA.

scriptors may signal importance; the full list is:

president, former, spokesman, sen, dr, chief, coach, attorney, minister, director, gov, rep, leader, secretary, rev, judge, US, general, manager, chairman.

Another lexical feature was the overall likelihood of a person's description using the bigram model from our web corpus. This indicates whether a person has a role or affiliation that is frequently mentioned. We performed 20-fold cross validation for both classification tasks. The results are shown in Table 2 (accuracy) and Table 3 (precision/recall).

4.1 Major vs Minor results

For major/minor classification, the majority class prediction has 94% accuracy, but is not a useful baseline as it predicts that *no* person should be mentioned by name and all are minor characters. J48 correctly predicts 114 major characters out of 258 in the 170 document sets. As recall appeared low, we further analyzed the 148 persons from DUC'03 and DUC'04 sets, for which DUC provides four human summaries. Table 4 presents the distribution of recall taking into account *how many* humans mentioned the person by name in their summary (originally, entities were labeled as main if *any* summary had a reference to them, cf. §3.2). It can be seen that recall is high (0.84) when all four humans consider a character to be major, and falls to 0.2 when only one out of four humans does. These observations reflect the well-known fact that humans differ in their choices for content selection, and indicate that in the automatic learning is more successful when there is more human agreement.

In our data there were 258 people mentioned by name in at least one human summary. In addition, there were 103 people who were mentioned in at least one human summary using only a common noun reference (these were identified by hand, as common noun coreference cannot be performed reliably enough by automatic means), indicating that 29% of people mentioned in human summaries are not actually named. Examples of such references include *an off duty black policeman*, *a Nigerian born Roman catholic priest*, *Kuwait's US ambassador*. For the purpose of generating references in a summary, it is important to evaluate how many of these people are correctly classified as minor characters. We removed these people from the training data and kept them as a test set. WEKA achieved a testing accuracy of 74% on these 103 test examples. But as discussed before, different human summarizers sometimes made different decisions on the form of reference to use. Out of the 103 referent for which a non-named reference was used by a summarizer, there were 40 where other summarizers used named reference. Only 22 of these 40 were labeled as minor characters in our automatic procedure. Out of the 63 people who were not named in *any* summary, but mentioned in at least one by common noun reference, WEKA correctly predicted 58 (92%) as minor characters. As before, we observe that when human summarizers generate references of the same form (reflecting consensus on conveying the perceived importance of the character), the machine predictions are accurate.

We performed feature selection to identify which are the most important features for the classification task. For the major/minor classification, the important features used by the classifier were the number of documents the person was mentioned in (feature 16), number of mentions within the document set (features 1,6), number of relative clauses (feature

4,5) and copula (feature 8) constructs, total number of apposition, relative clauses and copula (feature 9), number of high frequency premodifiers (feature 14) and the maximum bigram probability (feature 12). It was interesting that presence of apposition did not select for either major or minor class. It is not surprising that the frequency of mention within and across documents were significant features—a frequently mentioned entity will naturally be considered important for the news report. Interestingly, the syntactic form of the references was also a significant indicator, suggesting that the centrality of the character was signaled by the journalists by using specific syntactic constructs in the references.

	Major/Minor	Hearer New/Old
WEKA	0.96 (J48)	0.76 (SMO)
Majority class prediction	0.94	0.54

Table 2: Cross validation *testing* accuracy results.

	Class	Precision	Recall	F-measure
SMO	hearer-new	0.84	0.68	0.75
	hearer-old	0.69	0.85	0.76
J48	major-character	0.85	0.44	0.58
	minor-character	0.96	0.99	0.98

Table 3: Cross validation *testing* P/R/F results.

Number of summaries containing the person	Number of examples	Number and % recalled by J48
1 out of 4	59	15 (20%)
2 out of 4	35	20 (57%)
3 out of 4	29	23 (79%)
4 out of 4	25	21 (84%)

Table 4: J48 Recall results and human agreement.

4.2 Hearer Old vs New Results

The majority class prediction for the hearer-old/new classification task is that no one is known to the reader and it leads to overall classification accuracy of 54%. Using this prediction in a summarizer would result in excessive detail in referring expressions and a consequent reduction in space available to summarize the news events. The SMO prediction outperformed the baseline accuracy by 22% and is more meaningful for real tasks.

For the hearer-old/new classification, the feature selection step chose the following features: the number of appositions (features 2,3) and relative clauses (feature 5), number of mentions within the document set (features 0,1), total number of apposition, relative clauses and copula (feature 10), number of high frequency premodifiers (feature 14) and the minimum bigram probability (feature 13). As in the minor-major classification, the syntactic choices for reference realization were useful features.

We conducted an additional experiment to see how the hearer old/new status impacts the use of apposition or relative clauses for elaboration in references produced in human summaries. It has been observed (Siddharthan et al., 2004) that on average these constructs occur 2.3 times *less* frequently in human summaries than in machine summaries. As we show, the use of postmodification to elaborate relates to the hearer-old/new distinction.

To determine when an appositive or relative clause can be used to modify a reference, we considered the 151 examples out of 258 where there was at least one relative clause or apposition describing the person in the input. We labeled an example as positive if *at least* one human summary contained an apposition or relative clause for that person and negative otherwise. There were 66 positive and 85 negative examples. This data was interesting because while for the majority of examples (56%) all the human summarizers agreed not to use postmodification, there were very few examples (under 5%) where all the humans agreed to postmodify. Thus it appears that for around half the cases, it should be obvious that no postmodification is required, but for the other half, human decisions go either way.

Notably, none of the hearer-old persons (using test predictions of SMO) were postmodified. Our cognitive status predictions cleanly partition the examples into those where postmodification is not required, and those where it might be. Since no intuitive rule handled the remaining examples, we added the testing predictions of hearer-old/new and major/minor as features to the list in Table 1, and tried to learn this task using the tree-based learner J48. We report a testing accuracy of 71.5% (majority class baseline is 56%). There were only three useful features—the predicted hearer-new/old status, the number of high frequency premodifiers for that person in the input (feature 14 in table 1) and the average number of postmodified initial references in the input documents (feature 17).

5 Validating the results on current news

We tested the classifiers on data different from that provided by DUC, and also tested human consensus on the hearer-new/old distinction. For these purposes, we downloaded 45 clusters from one day's output from Newsblaster[4]. We then automatically compiled the list of people mentioned in the machine summaries for these clusters. There were 107 unique people that appeared in the machine summaries, out of 1075 people in the input clusters.

5.1 Human agreement on hearer-old/new

A question arises when attempting to infer hearer-new/old status: Is it meaningful to generalize this across readers, seeing how dependent it is on the world knowledge of individual readers?

To address this question, we gave 4 American graduate students a list of the names of people in the DUC human summaries (cf. §3), and asked them to write down for each person, their country/state/organization affiliation and their role (writer/president/attorney-general etc.). We considered a person hearer-old to a subject if they correctly identified both role and affiliation for that person. For the 258 people in the DUC summaries, the four subjects demonstrated 87% agreement ($\kappa = 0.74$)[5].

Similarly, they were asked to perform the same task for the Newsblaster data, which dealt with contemporary news[6], in contrast with the DUC data that contained news from the the late 80s and early 90s. On this data, the human agreement was 91% ($\kappa = 0.78$). This is a high enough agreement to suggest that the classification of national and international figures as hearer old/new across *the educated adult American reader with varied interests and background in current and recent events* is a well defined task. This is not necessarily true for the full range of cognitive status distinctions; for example Poesio and Vieira (1998) report lower human agreement on more fine-grained classifications of definite descriptions.

5.2 Results on the Newsblaster data

We measured how well the models trained on DUC data perform with current news labeled using human judgment. For each person who was mentioned in the automatic summaries for the Newsblaster data, we compiled one judgment from the 4 human subjects: an example was labeled as hearer-new if two or more out of the four subjects had marked it as hearer new. Then we used this data as *test data*, to test the model trained solely on the DUC data. The classifier for hearer-old/hearer-new distinction achieved 75% accuracy on Newsblaster data labeled by humans, while the cross-validation accuracy on the automatically labeled DUC data was 76%. These numbers are very encouraging, since they indicate that the performance of the classifier is stable and does not vary between the DUC and Newsblaster data. The precision and recall for the Newsblaster data are also very similar for those obtained from cross-validation on the DUC data:

Class	Precision	Recall	F-Measure
Hearer-old	0.88	0.73	0.80
Hearer-new	0.57	0.79	0.66

5.3 Major/Minor results on Newsblaster data

For the Newsblaster data, no human summaries were available, so no direct indication on whether a human summarizer will mention a person in a summary was available. In order to evaluate the performance of the classifier, we gave a human annotator the list of people's names appearing in the machine summaries, together with the input cluster and the machine summary, and asked which of the names on the list would be a suitable keyword for the set (keyword lists are a form of a very short summary). Out of the 107 names on the list, the annotator chose 42 as suitable for descriptive keyword for the set.

The major/minor classifier was run on the 107 examples; only 40 were predicted to be major characters. Of the 67 test cases that were predicted by the classifier to be minor characters, 12 (18%) were marked by the annotator as acceptable keywords. In comparison, of the 40 characters that were predicted to be major characters by the classifier, 30 (75%) were marked as possible keywords. If the keyword selections of the annotator are taken as ground truth, the automatic predictions have precision and recall of 0.75 and 0.71 respectively for the *major* class.

6 Conclusions

Cognitive status distinctions are important when generating summaries, as they help determine both

[4]`http://newsblaster.cs.columbia.edu`

[5]κ (kappa) is a measure of inter-annotator agreement over and above what might be expected by pure chance (See Carletta (1996) for discussion of its use in NLP). $\kappa = 1$ if there is perfect agreement between annotators and $\kappa = 0$ if the annotators agree only as much as you would expect by chance.

[6]The human judgments were made within a week of the news stories appearing.

what to say and how to say it. However, to date, no one has attempted the task of inferring cognitive status from unrestricted news.

We have shown that the hearer-old/new and major/minor distinctions can be inferred using features derived from the lexical and syntactic forms and frequencies of references in the news reports. We have presented results that show agreement on the *familiarity* distinction between educated adult American readers with an interest in current affairs, and that the learned classifier accurately predicts this distinction. We have demonstrated that the acquired cognitive status is useful for determining which characters to name in summaries, and which named characters to describe or elaborate. This provides the foundation for a principled framework in which to address the question of how much references can be shortened without compromising readability.

References

R. Barzilay. 2003. *Information Fusion for Multidocument Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University, New York.

D. Bikel, R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231.

J. Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

H. Daumé III, A. Echihabi, D. Marcu, D.S. Munteanu, and R. Soricut. 2002. GLEANS: A generator of logical extracts and abstracts for nice summaries. In *Proceedings of the Second Document Understanding Conference (DUC 2002)*, pages 9 – 14, Philadelphia, PA.

P. Gordon, B. Grosz, and L. Gilliom. 1993. Pronouns, names, and the centering of attention in discourse. *Cognitive Science*, 17:311–347.

H.P. Grice. 1975. Logic and conversation. In P. Cole and J.L. Morgan, editors, *Syntax and semantics*, volume 3, pages 43–58. Academic Press.

B. Grosz and C. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 3(12):175–204.

B. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

C. Grover, C. Matheson, A. Mikheev, and M. Moens. 2000. Lt ttt: A flexible tokenization toolkit. In *Proceedings of LREC'00*.

J. Gundel, N. Hedberg, and R. Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

Y. Guo, X. Huang, and L. Wu. 2003. Approaches to event-focused summarization based on named entities and query words. In *Document Understanding Conference (DUC'03)*.

K. Knight and D. Marcu. 2000. Statistics-based summarization — step one: Sentence compression. In *Proceeding of The American Association for Artificial Intelligence Conference (AAAI-2000)*, pages 703–710.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

I. Mani and M. Maybury, editors. 1999. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, Massachusetts.

A. Nenkova and K. McKeown. 2003. References to named entities: a corpus study. In *Proceedings of HLT/NAACL 2003*.

C. D. Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Inf. Process. Manage.*, 26(1):171–186.

M. Poesio and R. Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

E. Prince. 1992. The zpg letter: subject, definiteness, and information status. In S. Thompson and W. Mann, editors, *Discourse description: diverse analyses of a fund raising text*, pages 295–325. John Benjamins.

D. Radev and K. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

H. Saggion and R. Gaizaukas. 2004. Multi-document summarization by cluster/profile relevance and redundancy removal. In *Document Understanding Conference (DUC04)*.

A. Sanford, K. Moar, and S. Garrod. 1988. Proper names as controllers of discourse focus. *Language and Speech*, 31(1):43–56.

A. Siddharthan, A. Nenkova, and K. McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 896–902, Geneva, Switzerland.

A. Siddharthan. 2003. *Syntactic simplification and Text Cohesion*. Ph.D. thesis, University of Cambridge, UK.

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

Bayesian Learning in Text Summarization

Tadashi Nomoto
National Institute of Japanese Literature
1-16-10 Yutaka Shinagawa
Tokyo 142-8585 Japan
nomoto@acm.org

Abstract

The paper presents a Bayesian model for text summarization, which explicitly encodes and exploits information on how human judgments are distributed over the text. Comparison is made against non Bayesian summarizers, using test data from Japanese news texts. It is found that the Bayesian approach generally leverages performance of a summarizer, at times giving it a significant lead over non-Bayesian models.

1 Introduction

Consider figure 1. What is shown there is the proportion of the times that sentences at particular locations are judged as relevant to summarization, or worthy of inclusion in a summary. Each panel shows judgment results on 25 Japanese texts of a particular genre; columns (G1K3), editorials (G2K3) and news stories (G3K3). All the documents are from a single Japanese news paper, and judgments are elicited from some 100 undergraduate students. While more will be given on the details of the data later (Section 3.2), we can safely ignore them here.

Each panel has the horizontal axis representing location or order of sentence in a document, and the vertical axis the proportion of the times sentences at particular locations are picked as relevant to summarization. Thus in G1K3, we see that the first sentence (to appear in a document) gets voted for about 12% of the time, while the 26th sentence is voted for less than 2% of the time.

Curiously enough, each of the panels exhibits a distinct pattern in the way votes are spread across a document: G1K3 has the distribution of votes (DOV) with sharp peaks around 1 and 14; in G2K3, the distribution is peaked around 1, with a small bump around 19; in G3K3, the distribution is sharply skewed to the left, indicating that the majority of votes went to the initial section of a document. What is interesting about the DOV is that we could take it as indicating a collective preference for what to extract for a summary. A question is then, can we somehow exploit the DOV in summarization? To our knowledge, no prior work seems to exist that addresses the question. The paper discusses how we could do this under a Bayesian modeling framework, where we explicitly represent and make use of the DOV by way of Dirichlet posterior (Congdon, 2003).[1]

2 Bayesian Model of Summaries

Since the business of extractive summarization, such as one we are concerned with here, is about ranking sentences according to how useful/important they are as part of summary, we will consider here a particular ranking scheme based on the probability of a sentence being part of summary under a given DOV, i.e.,

$$P(y|\boldsymbol{v}), \tag{1}$$

where y denotes a given sentence, and $\boldsymbol{v} = (v_1, \ldots, v_n)$ stands for a DOV, an array of observed vote counts for sentences in the text; v_1 refers to the count of votes for a sentence at the text initial position, v_2 to that for a sentence occurring at the second place, etc.

Thus given a four sentence long text, if we have three people in favor of a lead sentence, two in favor

[1]See Yu et al. (2004) and Cowans (2004) for its use in IR.

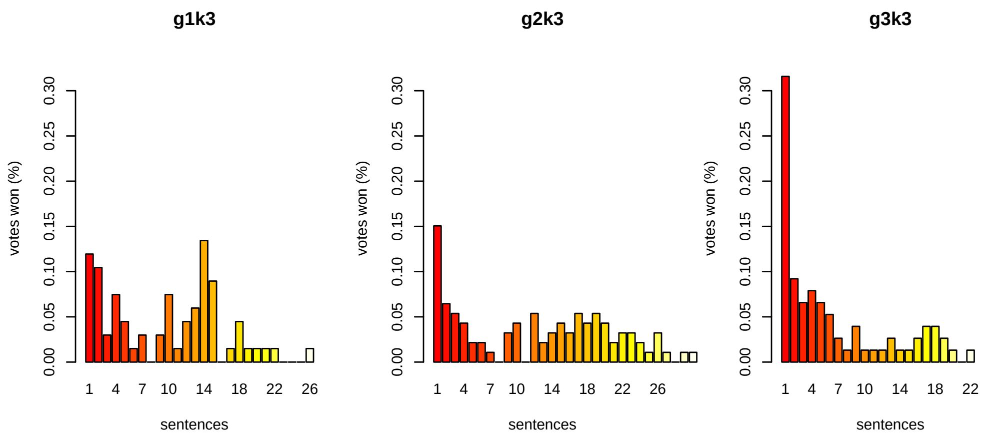


Figure 1: Genre-by-genre vote distribution

of the second, one for the third, and none for the fourth, then we would have $\boldsymbol{v} = (3, 2, 1, 0)$.

Now suppose that each sentence y_i (i.e., a sentence at the i-th place in the order of appearance) is associated with what we might call a prior preference factor θ_i, representing how much a sentence at a particular position is favored as part of a summary in general. Then the probability that y_i finds itself in a summary is given as:

$$\phi(y_i|\theta_i)P(\theta_i), \tag{2}$$

where ϕ denotes some likelihood function, and $P(\theta_i)$ a prior probability of θ_i.

Since the DOV is something we could actually observe about θ_i, we might as well couple θ_i with $\boldsymbol{v}$ by making a probability of θ_i conditioned on $\boldsymbol{v}$. Formally, this would be written as:

$$\phi(y_i|\theta_i)P(\theta_i|\boldsymbol{v}). \tag{3}$$

The problem, however, is that we know nothing about what each θ_i looks like, except that it should somehow be informed by $\boldsymbol{v}$. A typical Bayesian solution to this is to 'erase' θ_i by marginalizing (summing) over it, which brings us to this:

$$P(y_i|\boldsymbol{v}) = \int \phi(y_i|\theta_i)P(\theta_i\,|\boldsymbol{v})\, d\theta_i. \tag{4}$$

Note that equation 4 no longer talks about the probability of y_i under a particular θ_i; rather it talks about the expected probability for y_i with respect to a preference factor dictated by $\boldsymbol{v}$. All we need to know about $P(\theta_i|\boldsymbol{v})$ to compute the expectation is $\boldsymbol{v}$ and a probability distribution P, and not θ_i's, anymore.

$$\begin{tikzcd} \boldsymbol{v} \arrow[r] \arrow[rr, bend left] & \boldsymbol{\theta} \arrow[r] & y_i \end{tikzcd}$$

Figure 2: A graphical view

We know something about $\boldsymbol{v}$, and this would leave us P. So what is it? In principle it could be any probability distribution. However largely for the sake of technical convenience, we assume it is one component of a multinomial distribution known as the Dirichlet distribution. In particular, we talk about Dirichlet($\boldsymbol{\theta}|\boldsymbol{v}$), namely a Dirichlet posterior of θ, given observations $\boldsymbol{v}$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_i, \ldots, \theta_n)$, and $\sum_i^n \theta_i = 1$ $(\theta_i > 0)$. (Remarkably, if $P(\theta)$ is a Dirichlet, so is $P(\theta|\boldsymbol{v})$.) $\boldsymbol{\theta}$ here represents a vector of preference factors for n sentences — which constitute the text.[2]

Accordingly, equation 4 could be rewritten as:

$$P(y_i|\boldsymbol{v}) = \int \phi(y_i|\boldsymbol{\theta})P(\boldsymbol{\theta}\,|\boldsymbol{v})\, d\boldsymbol{\theta}. \tag{5}$$

An interesting way to look at the model is by way of a graphical model (GM), which gives some intuitive idea of what the model looks like. In a GM perspective, our model is represented as a simple tripartite structure (figure 2), in which each node corresponds to a variable (parameter), and arcs represent

[2]Since texts generally vary in length, we may set n to a sufficiently large number so that none of texts of interest may exceed it in length. For texts shorter than n, we simply add empty sentences to make them as long as n.

dependencies among them. $x \rightarrow y$ reads 'y depends on x.' An arc linkage between $\boldsymbol{v}$ and y_i is meant to represent marginalization over $\boldsymbol{\theta}$.

Moreover, we will make use of a scale parameter $\lambda \geq 1$ to have some control over the shape of the distribution, so we will be working with Dirichlet($\theta|\lambda\boldsymbol{v}$) rather than Dirichlet($\theta|\boldsymbol{v}$). Intuitively, we might take λ as representing a degree of confidence we have in a set of empirical observations we call $\boldsymbol{v}$, as increasing the value of λ has the effect of reducing variance over each θ_i in θ.

The expectation and variance of Dirichlet($\boldsymbol{\theta}|\boldsymbol{v}$) are given as follows.[3]

$$E[\theta_i] = \frac{v_i}{v_0} \tag{6}$$

$$Var[\theta_i] = \frac{v_i(v_0 - v_i)}{v_0^2(v_0 + 1)}, \tag{7}$$

where $v_0 = \sum_i^n v_i$. Therefore the variance of a scaled Dirichlet is:

$$Var[\theta_i|\lambda\boldsymbol{v}] = \frac{v_i(v_0 - v_i)}{v_0^2(\lambda v_0 + 1)}. \tag{8}$$

See how λ is stuck in the denominator. Another obvious fact about the scaling is that it does not affect the expectation, which remains the same.

To get a feel for the significance of λ, consider figure 3; the left panel shows a histogram of 50,000 variates of p_1 randomly drawn from Dirichlet($p_1, p_2|\lambda c_1, \lambda c_2$), with $\lambda = 1$, and both c_1 and c_2 set to 1. The graph shows only the p_1 part but things are no different for p_2. (The x-dimension represents a particular value p_1 takes (which ranges between 0 and 1) and the y-dimension records the number of the times p_1 takes that value.) We see that points are spread rather evenly over the probability space. Now the right panel shows what happens if you increase λ by a factor of 1,000 (which will give you $P(p_1, p_2|1000, 1000)$); points take a bell shaped form, concentrating in a small region around the expectation of p_1. In the experiments section, we will return to the issue of λ and discuss how it affects performance of summarization.

Let us turn to the question of how to find a solution to the integral in equation 5. We will be concerned here with two standard approaches to the issue: one is based on MAP (maximum a posteriori) and another on numerical integration. We start off with a MAP based approach known as Bayesian Information Criterion or BIC.

For a given model m, BIC seeks an analytical approximation for equation 4, which looks like the following:

$$\ln P(y_i|m) = \ln \phi(y_i|\hat{\boldsymbol{\theta}}, m) - \frac{k}{2} \ln N, \tag{9}$$

where k denotes the number of free parameters in m, and N that of observations. $\hat{\boldsymbol{\theta}}$ is a MAP estimate of $\boldsymbol{\theta}$ under m, which is $E[\boldsymbol{\theta}]$. It is interesting to note that BIC makes no reference to prior. Also worthy of note is that a minus of BIC equals MDL (Minimum Description Length).

Alternatively, one might take a more straightforward (and fully Bayesian) approach known as the Monte Carlo integration method (MacKay, 1998) (MC, hereafter) where the integral is approximated by:

$$P(y_i|\boldsymbol{v}) \approx \frac{1}{n} \sum_{j=1}^{n} \phi(y_i|x^{(j)}), \tag{10}$$

where we draw each sample $x^{(j)}$ randomly from the distribution $P(\boldsymbol{\theta}|\boldsymbol{v})$, and n is the number of $x^{(i)}$'s so collected. Note that MC gives an expectation of $P(y_i|\boldsymbol{v})$ with respect to $P(\boldsymbol{\theta}|\boldsymbol{v})$.

Furthermore, ϕ could be any probabilistic function. Indeed any discriminative classifier (such as C4.5) will do as long as it generates some kind of probability. Given ϕ, what remains to do is essentially training it on samples bootstrapped (i.e., resampled) from the training data based on $\boldsymbol{\theta}$ — which we draw from Dirichlet($\boldsymbol{\theta}|\boldsymbol{v}$).[4] To be more specific, suppose that we have a four sentence long text and an array of probabilities $\boldsymbol{\theta} = (0.4, 0.3, 0.2, 0.1)$ drawn from a Dirichlet distribution: which is to say, we have a preference factor of 0.4 for the lead sentence, 0.3 for the second sentence, etc. Then we resample with replacement lead sentences from training data with the probability of 0.4, the second with the probability of 0.3, and so forth. Obviously, a

[3]http://www.cis.hut.fi/ahonkela/dippa/dippa.html

[4]It is fairly straightforward to sample from a Dirichlet posterior by resorting to a gamma distribution, which is what is happening here. In case one is working with a distribution it is hard to sample from, one would usually rely on Markov chain Monte Carlo (MCMC) or variational methods to do the job.

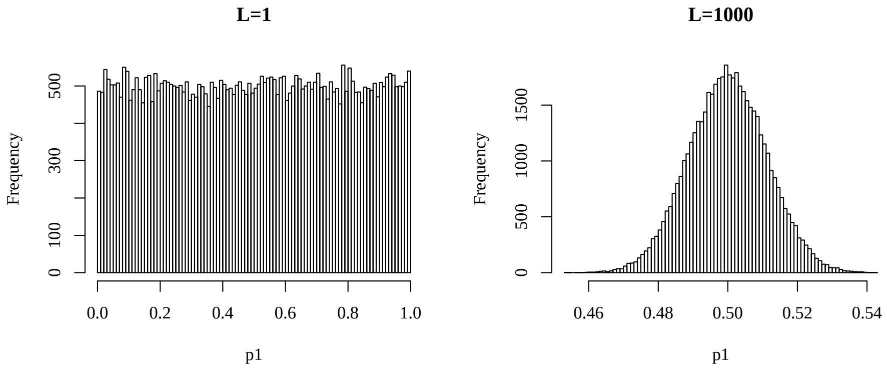


Figure 3: Histograms of random draws from Dirichlet$(p_1, p_2|\lambda c_1, \lambda c_2)$ with $\lambda = 1$ (left panel), and $\lambda = 1000$ (right panel).

high preference factor causes the associated sentence to be chosen more often than those with a low preference.

Thus given a text $T = (a, b, c, d)$ with $\boldsymbol{\theta} = (0.4, 0.3, 0.2, 0.1)$, we could end up with a data set dominated by a few sentence types, such as $T' = (a, a, a, b)$, which we proceed to train a classifier on in place of T. Intuitively, this amounts to inducing the classifier to attend to or focus on a particular region or area of a text, and dismiss the rest. Note an interesting parallel to boosting (Freund and Schapire, 1996) and the alternating decision tree (Freund and Mason, 1999).

In MC, for each $\boldsymbol{\theta}^{(k)}$ drawn from Dirichlet$(\boldsymbol{\theta}|\boldsymbol{v})$, we resample sentences from the training data using probabilities specified by $\boldsymbol{\theta}^{(k)}$, use them for training a classifier, and run it on a test document d to find, for each sentence in d, its probability of being a 'pick' (summary-worthy) sentence,i.e., $P(y_i|\boldsymbol{\theta}^{(k)})$, which we average across $\boldsymbol{\theta}$'s. In experiments later described, we apply the procedure for 20,000 runs (meaning we run a classifier on each of 20,000 $\boldsymbol{\theta}$'s we draw), and average over them to find an estimate for $P(y_i|\boldsymbol{v})$.

As for BIC, we generally operate along the lines of MC, except that we bootstrap sentences using only $E[\boldsymbol{\theta}]$, and the model complexity term, namely, $-\frac{k}{2}\ln N$ is dropped as it has no effect on ranking sentences. As with MC, we train a classifier on the bootstrapped samples and run it on a test document. Though we work with a set of fixed parameters, a bootstrapping based on them still fluctuates, producing a slightly different set of samples each time we run the operation. To get a reasonable convergence in experiments, we took the procedure to 5,000 iterations and averaged over the results.

Either with BIC or with MC, building a summarizer on it is a fairly straightforward matter. Given a document d and a compression rate r, what a summarizer would do is simply rank sentences in d based on $P(y_i|\boldsymbol{v})$ and pick an r portion of highest ranking sentences.

3 Working with Bayesian Summarist

3.1 C4.5

In what follows, we will look at whether and how the Bayesian approach, when applied for the C4.5 decision tree learner (Quinlan, 1993), leverages its performance on real world data. This means our model now operates either by

$$P(y_i|\boldsymbol{v}) \approx \frac{1}{n}\sum_{j=1}^{n}\phi_{c4.5}(y_i|x^{(j)}), \qquad (11)$$

or by

$$\ln P(y_i|m) = \ln \phi_{c4.5}(y_i|\hat{\boldsymbol{\theta}}, m) - \frac{k}{2}\ln N, \qquad (12)$$

with the likelihood function ϕ filled out by C4.5. Moreover, we compare two versions of the classifier; one with BIC/MC and one without. We used Weka implementations of the algorithm (with default settings) in experiments described below (Witten and Frank, 2000).

While C4.5 here is configured to work in a binary (positive/negative) classification scheme, we run it in a 'distributional' mode, and use a particular class membership probability it produces, namely, the probability of a sentence being positive, i.e., a pick (summary-worthy) sentence, instead of a category label.

Attributes for C4.5 are broadly intended to represent some aspects of a sentence in a document, an object of interest here. Thus for each sentence ψ, its encoding involves reference to the following set of attributes or features. 'LocSen' gives a normalized location of ψ in the text, i.e., a normalized distance from the top of the text; likewise, 'LocPar' gives a normalized location of the paragraph in which ψ occurs, and 'LocWithinPar' records its normalized location within a paragraph. Also included are a few length-related features such as the length of text and sentence. Furthermore we brought in some language specific feature which we call 'EndCue.' It records the morphology of a linguistic element that ends ψ, such as inflection, part of speech, etc.

In addition, we make use of the weight feature ('Weight') for a record on the importance of ψ based on tf.idf. Let $\psi = w_1, \ldots, w_n$, for some word w_i. Then the weight $W(\psi)$ is given as:

$$W(\psi) = \sum_{w} (1 + \log(\text{tf}(w))) \cdot \log(N/\text{df}(w)).$$

Here 'tf(w)' denotes the frequency of word w in a given document, 'df(w)' denotes the 'document frequency' of w, or the number of documents which contain an occurrence of w. N represents the total number of documents.[5]

Also among the features used here is 'Pos,' a feature intended to record the position or textual order of ψ, given by how many sentences away it occurs from the top of text, starting with 0.

While we do believe that the attributes discussed above have a lot to do with the likelihood that a given sentence becomes part of summary, we choose not to consider them parameters of the Bayesian model, just to keep it from getting unduly complex. Recall the graphical model in figure 2.

[5]Although one could reasonably argue for normalizing $W(\psi)$ by sentence length, it is not entirely clear at the moment whether it helps in the way of improving performance.

3.2 Test Data

Here is how we created test data. We collected three pools of texts from different genres, columns, editorials and news stories, from a Japanese financial paper (*Nihon Keizai Shinbun*) published in 1995, each with 25 articles. Then we asked 112 Japanese students to go over each article and identify 10% worth of sentences they find most important in creating a summary for that article. For each sentence, we recorded how many of the subjects are in favor of its inclusion in summary. On average, we had about seven people working on each text. In the following, we say sentences are 'positive' if there are three or more people who like to see them in a summary, and 'negative' otherwise. For convenience, let us call the corpus of columns G1K3, that of editorials G2K3 and that of news stories G3K3. Additional details are found in table 1.

4 Results and Discussion

Tables 2 through 4 show how the Bayesian summarist performs on G1K3, G2K3, and G3K3. The tables list results in precision at compression rates (r) of interest ($0 < r < 1$). The figures thereof indicate performance averaged over leave-one-out cross validation folds. What this means is that you leave out one text for testing and use the rest for training, which you repeat for each one of the texts in the data. Since we have 25 texts for each data set, this leads to a 25-fold cross validation. Precision is defined by the ratio of hits (positive sentences) to the number of sentences retrieved, i.e., r-percent of sentences in the text.[6]

In each table, figures to the left of the vertical line indicate performance of summarizers with BIC/MC and those to the right that of summarizers without them. Parenthetical figures like '(5K)' and '(20K)' indicate the number of iterations we took them to: thus BIC(5K) refers to a summarizer based on C4.5/BIC with scores averaged over 5,000 runs. BSE denotes a reference summarizer based on a regular C4.5, which it involves no resampling of training data. LEAD refers to a summarizer which works

[6]We do not use recall for a evaluation measure, as the number of positive instances varies from text to text, and may indeed exceed the length of a summary under a particular compression rate.

Table 1: N represents the number of sentences in G1K3 to G3K3. Sentences with three or more votes in their favor are marked positive, that is, for each sentence marked positive, at least three people are in favor of including it in a summary.

Genre	N	Positive (≥ 3)	Negative	P/N Ratio
G1K3	426	67	359	0.187
G2K3	558	93	465	0.200
G3K3	440	76	364	0.210

Table 2: G1K3. $\lambda = 5$. Dashes indicate no meaningful results.

r	BIC (5K)	MC (20K)	BSE	LEAD
0.05	0.4583	0.4583	–	0.3333
0.10	0.4167	0.4167	–	0.3472
0.15	0.3333	0.3472	–	0.2604
0.20	0.2757	0.2861	–	0.2306
0.25	0.2525	0.2772	–	0.2233
0.30	0.2368	0.2535	–	0.2066

Table 3: G2K3. $\lambda = 5$.

r	BIC (5K)	MC (20K)	BSE	LEAD
0.05	0.6000	0.5800	0.4200	0.5400
0.10	0.4200	0.4200	0.3533	0.3933
0.15	0.3427	0.3560	0.2980	0.3147
0.20	0.3033	0.3213	0.2780	0.2767
0.25	0.2993	0.2776	0.2421	0.2397
0.30	0.2743	0.2750	0.2170	0.2054

Table 4: G3K3. $\lambda = 5$.

r	BIC (5K)	MC (20K)	BSE	LEAD
0.05	0.9600	0.9600	0.8400	0.9600
0.10	0.7600	0.7600	0.6800	0.7000
0.15	0.6133	0.6000	0.5867	0.5133
0.20	0.5233	0.5233	0.4967	0.4533
0.25	0.4367	0.4367	0.3960	0.3840
0.30	0.4033	0.4033	0.3640	0.3673

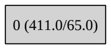


Figure 4: A non Bayesian C4.5 trained on G1K3.

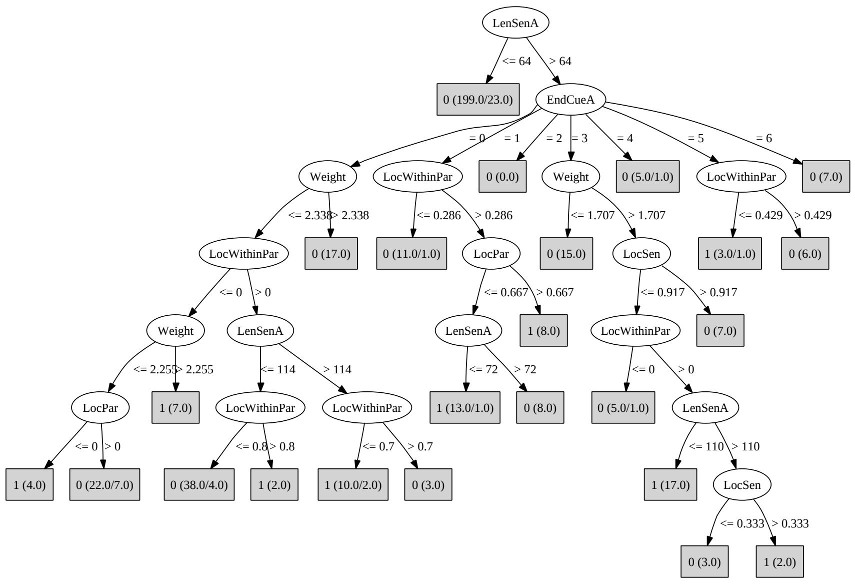


Figure 5: A Bayesian (MC) C4.5 trained on G1K3.

by selecting sentences from the top of the text. It is generally considered a hard-to-beat approach in the summarization literature.

Table 4 shows results for G3K3 (a news story domain). There we find a significantly improvement to performance of C4.5, whether it operates with BIC or MC. The effect is clearly visible across a whole range of compression rates, and more so at smaller rates.

Table 3 demonstrates that the Bayesian approach is also effective for G2K3 (an editorial domain), outperforming both BSE and LEAD by a large margin.

Similarly, we find that our approach comfortably beats LEAD in G1K3 (a column domain). Note the dashes for BSE. What we mean by these, is that we obtained no meaningful results for it, because we were unable to rank sentences based on predictions by BSE. To get an idea of how this happens, let us look at a decision tree BSE builds for G1K3, which is shown in figure 4. What we have there is a decision tree consisting of a single leaf.[7] Thus for whatever sentence we feed to the tree, it throws back the same membership probability, which is 65/411. But then this would make a BSE based summarizer utterly useless, as it reduces to generating a summary by picking at random, a particular portion of text.[8]

Now Figure 5 shows what happens with the Bayesian model (MC), for the same data. There we see a tree of a considerable complexity, with 24 leaves and 18 split nodes.

Let us now turn to the issues with λ. As we might recall, λ influences the shape of a Dirichlet distribution: a large value of λ causes the distribution to have less variance and therefore to have a more acute peak around the expectation. What this means is that increasing the value of λ makes it more likely to have us drawing samples closer to the expectation. As a consequence, we would have the MC model acting more like the BIC model, which is based on MAP estimates. That this is indeed the case is demonstrated by table 5, which gives results for the MC model on G1K3 to G3K3 at $\lambda = 1$. We see that the MC behaves less like the BIC at $\lambda = 1$ than at $\lambda = 5$ (table 2 through 4).

Of a particular interest in table 5 is G1K3, where the MC suffers a considerable degradation in performance, compared to when it works with $\lambda = 5$. G2K3 and G3K3, again, witness some degradation in performance, though not as extensive as in G1K3. It is interesting that at times the MC even works better with $\lambda = 1$ than $\lambda = 5$ in G2K3 and G3K3.[9]

[7]This is not at all surprising as over 80% of sentences in a non resampled text are negative for the most of the time.

[8]Its expected performance (averaged over 10^6 runs) comes to: 0.1466 ($r = 0.05$), 0.1453 ($r = 0.1$), 0.1508 ($r = 0.15$), 0.1530 ($r = 0.2$), 0.1534 ($r = 0.25$), and 0.1544 ($r = 0.3$).

[9]The results suggest that if one like to have some improvement, it is probably a good idea to set λ to a large value. But

Table 5: MC (20K). $\lambda = 1$.

r	G1K3	G2K3	G3K3
0.05	0.3333	0.5400	0.9600
0.10	0.3333	0.3867	0.7800
0.15	0.2917	0.3960	0.5867
0.20	0.2549	0.3373	0.5200
0.25	0.2480	0.2910	0.4347
0.30	0.2594	0.2652	0.4100

All in all, the Bayesian model proves more effective in leveraging performance of the summarizer on a DOV exhibiting a complex, multiply peaked form as in G1K3 and G2K3, and less on a DOV which has a simple, single-peak structure as in G3K3 (cf. figure 1).[10]

5 Concluding Remarks

The paper showed how it is possible to incorporate information on human judgments for text summarization in a principled manner through Bayesian modeling, and also demonstrated how the approach leverages performance of a summarizer, using data collected from human subjects.

The present study is motivated by the view that that summarization is a particular form of collaborative filtering (CF), wherein we view a summary as a particular set of sentences favored by a particular user or a group of users just like any other things people would normally have preference for, such as CDs, books, paintings, emails, news articles, etc. Importantly, under CF, we would not be asking, what is the 'correct' or gold standard summary for document X? – the question that consumed much of the past research on summarization. Rather, what we are asking is, what summary is popularly favored for X?

Indeed the fact that there could be as many summaries as angles to look at the text from may favor the CF view of summary: the idea of what constitutes a good summary may vary from person to person, and may well be influenced by particular interests and concerns of people we elicit data from.

Among some recent work with similar concerns, one notable is the Pyramid scheme (Nenkova and Passonneau, 2004) where one does not declare a particular human summary a absolute reference to compare summaries against, but rather makes every one of multiple human summaries at hand bear on evaluation; Rouge (Lin and Hovy, 2003) represents another such effort. The Bayesian summarist represents yet another, whereby one seeks a summary most typical of those created by humans.

in general how to best set λ requires some experimenting with data and the optimal value may vary from domain to domain. An interesting approach would be to empirically optimize λ using methods suggested in MacKay and Peto (1994).

[10]Incidentally, summarizers, Bayesian or not, perform considerably better on G3K3 than on G1K3 or G2K3. This happens presumably because a large portion of votes concentrate in a rather small region of text there, a property any classifier should pick up easily.

References

Peter Congdon. 2003. *Bayesian Statistical Modelling*. John Wiley and Sons.

Philip J. Cowans. 2004. Information Retrieval Using Hierarchical Dirichlet Processes. In *Proc. 27th ACM SIGIR*.

Yoav Freund and Llew Mason. 1999. The alternating decision tree learning algorithm,. In *Proc. 16th ICML*.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proc. 13th ICML*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurance statistics. In *Proc. HLT-NAACL 2003*.

David J. C. MacKay and Linda C. Bauman Peto. 1994. A Hierarchical Dirichlet Language Model. *Natural Language Engineering*.

D. J. C. MacKay. 1998. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, Kluwer Academic Press.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluation Content Selection in Summarization: The Pyramid Method. In *Proc. HLT-NAACL 2004*.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Kai Yu, Volker Tresp, and Shipeng Yu. 2004. A Nonparametric Hierarchical Bayesian Framework for Information Filtering. In *Proc. 27th ACM SIGIR*.

Discourse Chunking and its Application to Sentence Compression

Caroline Sporleder and **Mirella Lapata**
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK
{csporled,mlap}@inf.ed.ac.uk

Abstract

In this paper we consider the problem of analysing sentence-level discourse structure. We introduce discourse chunking (i.e., the identification of intra-sentential nucleus and satellite spans) as an alternative to full-scale discourse parsing. Our experiments show that the proposed modelling approach yields results comparable to state-of-the-art while exploiting knowledge-lean features and small amounts of discourse annotations. We also demonstrate how discourse chunking can be successfully applied to a sentence compression task.

1 Introduction

The computational treatment of discourse phenomena has recently attracted much attention, partly due to their increasing importance for potential applications. In summarisation, for example, the extraction of sentences to include in a summary crucially depends on their rhetorical status (Marcu, 2000; Teufel and Moens, 2002); one might want to extract contrastive or explanatory statements while omitting sentences that contain background information. In information extraction, discourse-level knowledge can be used to identify co-referring events (Humphreys et al., 1997) and to determine their temporal order. Discourse processing could further enhance question answering systems by interpreting the user's question either in isolation or in the context of preceding questions (Chai and Jing, 2004).

Discourse analysis is often viewed as a parsing task. Rhetorical Structure Theory (RST, Mann and Thomson, 1988), one of the most influential frameworks in discourse processing, represents texts by trees whose leaves correspond to elementary discourse units (*edus*) and whose nodes specify how these and larger units (e.g., multi-sentence segments) are linked to each other by rhetorical relations (e.g., *Contrast*, *Elaboration*). Discourse units are further characterised in terms of their text importance: *nuclei* denote central segments, whereas *satellites* denote peripheral ones.

Recent advances in discourse modelling have greatly benefited from the availability of resources annotated with discourse-level information such as the RST Discourse Treebank (RST-DT, Carlson et al., 2002). Even though discourse parsing at the document-level still poses a significant challenge to data-driven methods, sentence-level discourse models (e.g., Soricut and Marcu, 2003) trained on the RST-DT have attained accuracies comparable to human performance. The availability of discourse annotations is partly responsible for the success of these models. Another important reason is the development of robust syntactic parsers (e.g., Charniak, 2000) that can be used to provide critical structural and lexical information to the discourse parser. Unfortunately, discourse annotated corpora are largely absent for languages other than English. Furthermore, reliance on syntactic parsing renders discourse parsing practically impossible for languages for which state-of-the-art parsers are unavailable.

In this paper we propose discourse chunking as an alternative to discourse parsing. Analogous to sentence chunking, discourse chunking is an intermediate step towards full parsing. Following an RST-style analysis, we focus solely on two subtasks: (a) discourse segmentation, i.e., determining which word sequences form *edus* and (b) inferring whether these *edus* function as nuclei or satellites. The motivation for tackling these subtasks is two-fold. First, they are of crucial importance for full-scale discourse parsing. For example, Soricut and Marcu (2003) show that perfect discourse segmentation delivers an error reduction of 29% in the performance of their discourse parser. Second, some applications may not require full-scale discourse parsing. For example, it has been shown that nuclearity is important for summarisation, i.e., nuclei are more likely to be retained when summarising than satellites (Marcu, 2000). While nuclearity alone may not be sufficient for document summarisation (Marcu, 1998), such knowledge could prove useful at the sentence level, for example for producing sentence compressions.

The algorithms introduced in this paper are purposely knowledge-lean. We abstain from using syntactic parsers or semantic databases such as WordNet (Fellbaum, 1998), thus exploring the portability of our methods to languages for which such resources are not available. We employ lexical and low-level syntactic information (e.g., parts of speech, syntactic chunks) and show that the performance of our discourse chunker on the two subtasks (mentioned above) is comparable to that of a state-of-the-art sentence-level discourse parser (Soricut and Marcu, 2003). We also assess its application potential on a sentence compression task (Knight and Marcu, 2003).

2 Related Work

Initial work towards the development of discourse parsers has primarily relied on hand-crafted rules for specifying world knowledge or constraints on tree structures (e.g., Hobbs 1993). Recent work has seen the emergence of treebanks annotated with discourse structure, thus enabling the development of more robust, data-driven models. Marcu (2000) presents a shift-reduce parsing model that segments texts into *edus* and determines how they should be assembled into rhetorical structure trees. Soricut and Marcu (2003) introduce a syntax-based sentence-level discourse parser, which consists of two components: a statistical segmentation model and a parser working on the output of the segmenter. Both components are trained on the RST-DT and exploit lexical features as well as syntactic dominance features (which are taken from syntactic parse trees).

Given that discourse-level information plays an important role in human summarisation (Endres-Niggemeyer, 1998), it is not surprising that models of discourse structure have found use in automatic summarisation. For instance, Marcu (2000) proposes a summarisation algorithm that builds an RST tree for the entire text, and identifies its most important parts according to discourse salience.

Our work differs from previous approaches in two key respects. First, we do not attempt to produce a hierarchical discourse structure. We introduce discourse chunking, a less resource demanding task than full discourse parsing. We show that good chunking performance can be achieved with low-level information. Second, we apply our discourse chunker to sentence compression. Although previous approaches have utilised discourse information for document summarisation, its application to sentence condensation is novel to our knowledge.

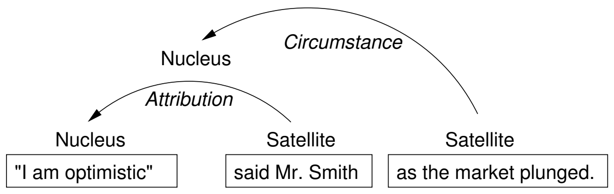


Figure 1: Discourse Tree in RST-DT

3 Discourse Chunking

3.1 Data and Representation

We propose a supervised machine learning approach to discourse chunking. Our data were obtained from the RST-DT (Carlson et al., 2002), which consists of 385 Wall Street Journal articles manually annotated with discourse structures in the framework of Mann and Thompson (1987). An example of an RST-based tree representation is shown in Figure 1; rectangular boxes denote *edus* and arcs indicate which relations (e.g., *Circumstance* or *Attribution*) hold between them. Relations are typically binary with one unit being the nucleus (indicated by arrows in Figure 1) and the other the satellite, but multi-nuclear and non-binary relations are also possible.

We are only interested in the lowest level of the tree, i.e., we aim to identify the *edus* and determine whether they are nuclei or satellites. For example, in the sentence in Figure 1 we want to identify the three *edus* *"I am optimistic"*, *said Mr. Smith*, and *as the market plunged.* and determine that the first of these functions as a nucleus at the lowest level of the tree whereas the latter two function as satellites. We do not try to determine that the first two *edus* are merged at a higher level and then function as the overall nucleus of the sentence.

The discourse chunking task assumes a non-hierarchical representation. We converted each sentence-level discourse tree into a flat chunk representation by assigning each token (i.e., word or punctuation mark) a tag encoding its nuclearity status at the *edu* level. We adopted the chunk representation proposed by Ramshaw and Marcus (1995) and used four different tags: B-NUC and B-SAT for nucleus and satellite-initial tokens, and I-NUC and I-SAT for non-initial tokens, i.e., tokens inside a nucleus and satellite span. As all tokens belong either to a nucleus or a satellite span, we do not need a special tag (typically denoted by O in syntactic chunking) to indicate elements outside a chunk. The chunk representation for the sentence in Figure 1 is thus:

"/B-NUC I/I-NUC am/I-NUC optimistic/I-NUC "/I-NUC said/B-SAT Mr./I-SAT Smith/I-SAT as/B-SAT the/I-SAT market/I-SAT plunged/I-SAT ./I-SAT

Discourse and sentence structure do not always correspond, and for 5% of sentences in the RST-DT no discourse tree exists. We excluded these from our data. We also disregarded sentences without internal structure, i.e., those which consist of only one *edu*. The RST-DT is partitioned into a training (342 articles) and test set (43 articles). We preserved this split in all our experiments. 52 articles in the RST-DT are doubly annotated. We used these to compute human agreement on the discourse chunking task (see Section 4.1).

3.2 Modelling

Using a chunk-based representation effectively renders discourse processing a sequence labelling task. Two modelling approaches are possible. The simplest model performs segmentation and labelling simultaneously. In our case this involves training a classifier that labels each token with one of our four tags (i.e., B-NUC, I-NUC, B-SAT, I-SAT). Alternatively, we could treat discourse chunking as two distinct subtasks involving two binary classifiers: a segmenter, which determines the chunk boundaries and assigns each token a chunk-initial (B) or non-chunk-initial tag (I), and a labeller, which classifies each chunk identified by the segmenter as either nucleus (NUC) or satellite (SAT).[1]

The second approach has a number of advantages. First, abstracting away from a token-based representation in the second step makes it easier to model sentence-level distributional properties of nuclei and satellites, e.g., the fact that every sentence has at least one nucleus. This can be achieved by incorporating additional features into the labeller, such as the number of chunks in the sentence or the length of the current chunk. A two-step approach also avoids the creation of illegal chunk sequences, such as "B-SAT I-NUC". However, a potential drawback is that the number of training examples for the labeller is reduced as the instances to be classified are chunks rather than tokens. We explore the performance of the one-step and the two-step methods in Sections 4.2 and 4.3, respectively.

[1]A similar approach has been proposed for syntactic chunking, e.g., Tjong Kim Sang (2000).

A variety of learning schemes can be employed for the discourse chunking task. We have experimented with Boosting (Schapire and Singer, 2000), Conditional Random Fields (Lafferty et al., 2001), and Support Vector Machines (Vapnik, 1998). Discussion of our results focuses exclusively on boosting, since it had a slight advantage over the other methods. Boosting combines many simple, moderately accurate categorisation rules into a single, highly accurate rule. We used BoosTexter's (Schapire and Singer, 2000) implementation, which combines boosting with simple decision rules. The system permits three different types of features: numeric, nominal and "text". Text-valued features can, for example, encode sequences of words or parts of speech. BoosTexter applies *n*-gram models when forming classification hypotheses for text-valued features.

3.3 Features for the Token-Based Models

While we use similar features for all our classifiers, their concrete implementation depends on whether the classifier is token-based (i.e., the one-step model and the segmenter in the two-step method) or span-based (i.e., the labeller in the two-step method). We first describe the features for the former.

Each token is represented as a feature vector encoding information about the token itself and its context. We intentionally limited our features to a basic set representing grammatical, syntactic, and lexical information.

Tokens This feature simply encodes the identity of the current token; we used raw tokens, without lemmatisation or stemming.

Part-of-Speech Tags Tokens were also annotated with parts of speech using a publicly available state-of-the-art tagger (Mikheev, 1997).

Syntactic Chunks Chunk information is a valuable cue for determining discourse segments; it is unlikely that a segment boundary occurs within a syntactic chunk. We applied a chunker (Mikheev, 1997) to our data to discover noun and verb phrase chunks. The chunker assigned one of five labels to each token, encoding the first element of a noun or verb chunk (B-NP and B-VP, respectively), a non-initial element in a chunk (I-NP and I-VP), and an element outside a chunk (O). We used these chunk labels directly as features and also encoded generalisations over chunk and boundary types (i.e., VP vs. NP and B vs. I, respectively).

Clause Information Knowing where clause boundaries lie is important for segmentation, since discourse segments often correspond to clauses. We used a rule-based algorithm (Leffa, 1998) to identify clauses from the syntactic chunker's output and recorded for every token whether it is clause-initial (S) or not (X).

Discourse Connectives Discourse connectives such as *but* often indicate which rhetorical relation holds between two spans. While we do not aim to infer the relation proper, knowing the type of relation holding between spans often helps in determining whether they should be labelled as nucleus or satellite. For example, *Contrast* relations (e.g., signalled by *but*) hold between two nuclei whereas *Cause* relations (e.g., signalled by *because*) hold between a nucleus and a satellite. Hence, we recorded the presence of discourse connectives in a sentence to capture, albeit in a shallow manner, the interdependency between rhetorical relations and nuclearity.

We used Knott's (1996) inventory of discourse connectives and encoded two types of information for each token: (a) whether the token is a connective (C) or not (X) and (b) the identity of the connective if the token is a connective (zero otherwise).[2]

Token Position For each token we calculated its relative position in the sentence (defined as the token position divided by the number of tokens). This information is useful to capture potential positional differences between nuclei and satellites, i.e., it may be that nuclei are more likely at the beginning of a sentence than at the end.

Context In addition to the nine features above, which encode information about the token itself, we also implemented 16 contextual features to encode information about its neighbouring tokens. Syntactic chunking approaches typically capture contextual information by defining a small window of a few tokens to the left and right of the current token (see Veenstra, 1998). However, we used the whole sentence as context, since BoosTexter is fairly good at determining automatically relevant *n*-grams within a longer string of tokens. We included this contextual information for all nominal features; that is, we encoded not only the string of preceding and following tokens but also the string of preceding and following part-of-speech tags, syntactic chunk labels, clause labels, and connectives. For example, we had three token features, one encoding the current token itself, and two contextual features (one encoding the string of preceding tokens, and one encoding the string of following tokens); similarly we had three part-of-speech features, nine syntactic chunk features three using the complete chunk tags, three using only the chunk type, and three using the boundary type), and so on.

[2]Some words can have syntactic as well as discourse marking functions (e.g., *but* sometimes functions as a synonym for *except* rather than as a *Contrast* marker). We do not disambiguate between these two usages.

3.4 Features for the Span-Based Model

For the labeller we encoded information about spans rather than tokens. This gave rise to six non-contextual, text-valued features: the string of tokens in the current span, their parts of speech, syntactic chunk tags, clause tags, and the presence and identity of connectives. The positional feature was redefined in terms of relative span position, i.e., the position of the current span divided by the number of spans in the sentence. We restricted contextual features to information about immediately preceding and following spans (within a sentence). We did not include information about non-adjacent spans because only a minority of sentences in our data contained more than three spans. Again, we included contextual information for all nominal features. Finally, to capture intra-sentential span-structure, we added the following features:

Span Length Span length was measured in terms of the number of tokens in it and was represented by three features: the length of the current span, and the lengths of its adjacent spans. Span length information captures differences in the average length of nuclei and satellite spans.

Number of Spans We encoded the number of spans in the sentence overall and the number of spans preceding and following the current span.

4 Experiments

In this section we describe the experiments that assess the merits of the discourse chunking framework introduced above. We also give details regarding parameter estimation and training for our models and introduce the baseline and state-of-the-art methods used for comparison with our approach.

4.1 Upper Bound

Before presenting the results of our modelling experiments, it is worth considering how well humans agree on discourse chunk segmentation and labelling in order to establish an upper bound for the task. We measured both unlabelled and labelled agreement on the 52 doubly annotated RST-DT texts. The former measures whether humans agree in placing chunk boundaries, whereas the latter additionally measures whether humans agree in assigning chunk labels. To facilitate comparison with our models we report inter-annotator agreement in terms of accuracy and F-score.[3] For the unlabelled case we also report *Window Difference* (*WDiff*), a commonly used evaluation measure for segmentation tasks (Pevzner and Hearst, 2002). It returns values between 0 (identical segmentations) and 1 (maximally different segmentations) and differs from accuracy in that predicted boundaries which are only slightly off are penalised less than those which are completely wrong.

Human agreement is relatively high[4] on both segmentation and span labelling (see Table 1), which can be explained by the fact that (i) the RST-DT annotators were given very detailed and precise instructions and (ii) assigning boundaries and labels is an easier task than creating full-scale discourse trees.

4.2 One-Step Chunking

For the one-step chunking method, our training set consists of approximately 130,000 instances (i.e., tokens). We set aside 10% as a development set for optimising BoosTexter's parameters (i.e., the number of training iterations and the maximal length of the n-grams considered for text-valued features). We then re-trained BoosTexter with the optimal setting (700 iterations, $n = 2$) and applied it to the test set, which contained around 15,500 instances.

By default, the one-step method treats every token in isolation, i.e., it assigns each token a tag without taking its neighbouring tags into account. This is not an entirely adequate model, since the likelihood of a tag is influenced by its surrounding tags. For example, the probability of a token being tagged as I-NUC should increase if the preceding token was tagged as B-NUC. One way to take information about surrounding tags into account is by stacking classifiers, i.e., adding the output of one classifier to the input of another. Stacking is frequently used in chunking tasks (e.g., Veenstra, 1998). We stack two BoosTexter classifiers, by adding the string of all preceding and following tags (within a given sentence) to each token's feature vector for the second classifier.

It would be possible to generate training material for the second classifier directly from the original training set by using the gold standard output tags in the augmented feature vector. However, we found that this leads BoosTexter to rely too much on these tags, largely ignoring other features. This causes problems when the model is applied to the test set where the class tags are predicted and may contain errors. Hence, we applied the original model (BT-1-Step) to obtain predicted output tags for the training data and then used these, rather than the gold standard tags, to train the second classifier. Similarly, during testing, we first applied BT-1-Step and used its output tags to augmented the feature vectors of the second classifier.

For comparison, we also applied two baseline models to our data. The first (BaseMaj) is obtained by always assigning the tag that is most common in the training data (I-NUC). This strategy makes no attempt at guessing span boundaries. The second (BaseClMaj) indirectly assesses the importance of clause boundary detection. It implements a strategy which assumes that span boundaries always coincide with clause boundaries. To obtain clause boundaries, we used the gold standard annotation of our data in the Penn Treebank. We then labelled all clause-initial tokens as B-NUC and all other tokens as I-NUC. Note, that the use of gold standard clause boundaries makes this a relatively high baseline. We also applied Spade[5], Soricut and Marcu's (2003) sentence-level discourse parser (see Section 2) to our test set. For evaluation purposes, Spade's output was converted to our chunk representation. It is important to note that Spade is a much more sophisticated model than the ones presented in this paper. We therefore do not expect to be able to obtain a better performance. It is nevertheless interesting to see how far one can go with a modest feature space and considerably less structural information.

Table 1 shows the results. A set of diacritics is used to indicate significance (on accuracy) throughout this paper, see Table 2. On the segmentation task (unlabelled) BT-1-Step and its stacked variant significantly outperform the majority baseline (BaseMaj) but are significantly less accurate than BaseClMaj, which uses gold standard clause boundaries. The two BoosTexter models also perform significantly worse than Spade on segmentation. However, the higher WDiff for Spade on the segmentation task suggests that the boundaries predicted by our models contain more "near misses" than those predicted by Spade. When segmentation and span labelling are taken into account (labelled), our one-step models significantly outperform both baselines but are significantly less accurate than Spade. Classifier stack-

[3]For the unlabelled case, we report the F-score on boundaries; for the labelled case, we report the average F-score over all class labels weighted by class frequency in the training set.

[4]Using the Kappa statistic agreement on segmentation is $K = .97$ and on span labelling $K = .81$.

[5]The software is publicly available from `http://www.isi.edu/licensed-sw/spade/`.

	unlabelled			labelled	
Models	Acc %	F-score	WDiff	Acc %	F-score
BaseMaj	88.50	–	.4021	53.87	38.77
BaseClMaj	93.51	70.06	.2008	56.64	43.62
BT-1-Step	90.07$\ast\dagger\ddagger\$$	64.64	.2148	74.40$\ast\dagger\ddagger\$$	74.13
BT-1-Step, stacked	91.86$\ast\not\dagger\ddagger\$$	68.95	.1795	75.55$\ast\dagger\ddagger\$$	75.37
BT-2-Step	97.37$\ast\dagger\ddagger\$$	88.28	.0733	78.27$\ast\dagger\not\ddagger\$$	78.38
BT-2-Step, stacked	97.41$\ast\dagger\ddagger\$$	88.40	.0727	76.31$\ast\dagger\ddagger\$$	76.34
Spade	93.49$\ast\not\dagger\$$	87.06	.5071	79.21$\ast\dagger\$$	80.91
Humans	99.05	97.96	.0012	89.10	89.03

Table 1: Results on discourse segmentation and span labelling

Symbols	Meaning
$\ast$ $\not\ast$	(not) sig different from BaseMaj
$\dagger$ $\not\dagger$	(not) sig different from BaseClMaj
$\ddagger$ $\not\ddagger$	(not) sig different from Spade
$\$$ $\not\$$	(not) sig different from Humans

Table 2: Meaning of diacritics indicating statistical significance (χ^2 tests, $p < 0.05$)

ing leads to slight improvements over the simple BoosTexter model, but the difference is not statistically significant.

4.3 Two-Step Chunking

In the two-step model, chunking consists of two separate subtasks: segmentation and labelling. To generate training material for the segmenter, we replaced the four chunk labels in the original data set by their corresponding boundary labels (B, I). For the labeller, training instances are spans rather than tokens. We used the gold standard span boundaries to convert the original training set to a span-based representation. This new training set contained around 15,000 instances (compared to 130,000 instances in the token-based set). For both the segmenter and labeller, we set aside 10% of the material as development data to optimise BoosTexter's parameters (900 iterations, $n = 3$ for segmentation, and 600 iterations, $n = 2$ for labelling).

For testing, we first applied the segmenter to obtain discourse chunk boundaries. We then used the predicted boundaries to convert the test data into a span-based representation, which we then used as input for the labeller. For evaluation, the output of the labeller was converted back to a token-based representation. As with one-step chunking, we also implemented a stacked variant, stacking both the segmentation and the labelling models.

It can be seen in Table 1 that the two-step models outperform the one-step models. This difference is significant except for the stacked model on the labelling task (labelled). Both two-step models significantly outperform both baselines on segmentation (unlabelled) and labelling (labelled). They also significantly outperform Spade on the boundary prediction task, which is in itself an important subtask for discourse parsing. The unstacked two-step BoosTexter model performs comparably to Spade with respect to labelled accuracy; the difference between the two models is not statistically significant. Hence, we achieve results similar to Spade but with much simpler and knowledge-leaner features. As with the one-step method, the stacked model performs (insignificantly) better than its unstacked counterpart on the segmentation task. However, on the labelling task, the stacked variant performs significantly worse. We conjecture that the reduced training set size for the labeller causes the stacked model (which is effectively trained twice) to overfit. Expectedly, all models perform significantly worse than humans on both tasks.

To assess whether our discourse chunker could be ported to languages for which discourse treebanks are not yet available, we investigated how much annotated data is required to achieve satisfactory results. Assuming that annotators proceed sentence-by-sentence, we varied the amount of sentences in our training data and determined its effect on the learner's (BT-2-Step) performance. Figure 2 shows that satisfactory labelled and unlabelled performance (86.52% and 74.64% F-score, respectively) can be achieved with approximately half the training data (i.e., around 2,000 sentences). In fact, using the entire data set yields a moderate increase of 1.78% for the unlabelled task and 3.68% for the labelled task. Hence, it seems that our knowledge-lean method is suitable even for relatively small training sets. We next examine whether the two-step chunking model can be usefully employed in a practical application such as sentence compression.

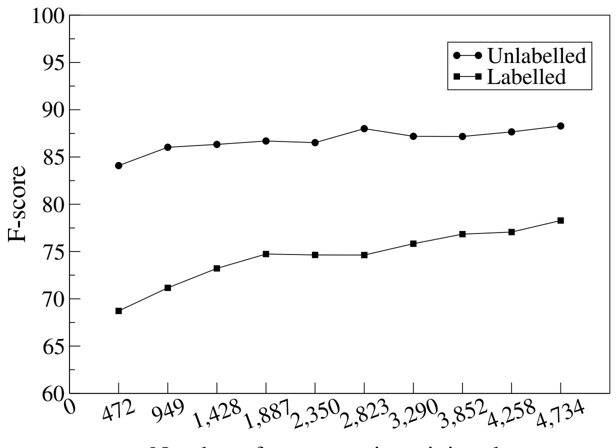


Figure 2: Learning curve for discourse segmentation (unlabelled) and span labelling (labelled)

4.4 Sentence Compression

Sentence compression can be likened to summarisation at the sentence level. The task has an immediate impact on several applications ranging from summarisation to audio scanning devices for the blind and caption generation (see Knight and Marcu, 2002 and the references therein). Previous data-driven approaches (Knight and Marcu, 2003; Riezler et al., 2003) relied on parallel corpora to determine what is important in a sentence. The models learned correspondences between long sentences and their shorter counterparts, typically employing a rich feature space induced from parse trees. The task is challenging since the compressed sentences should retain essential information and convey it grammatically.

Here, we propose a complementary approach which utilises discourse chunking. A compressed sentence can be obtained from the output of the chunker simply by removing satellites. We thus capitalise on RST's (Mann and Thompson, 1987) notion of nuclearity and the widely held assumption that spans functioning as satellites can often be deleted without disrupting coherence. To evaluate the compressions produced by our chunking model, we elicited judgements from human subjects. We describe our elicitation study and results as follows.

Data We randomly selected 40 sentences from the test portion of the RST-DT. Average sentence length was 38.75. The sentences were compressed by chunking them with our (unstacked) two-step model (BT-2-Step) and then dropping satellites. We applied the same strategy to derive compressed sentences from the output of Spade (Soricut and Marcu, 2003), and also produced human compressions. Finally, we added a simple baseline compression algorithm proposed by Jing and McKeown (2000) which removed all prepositional phrases, clauses, to-infinitives, and gerunds. Both the baseline and Spade operate on parse trees which were obtained from Charniak's (2000) parser. Our set of experimental materials contained $4 \times 40 = 160$ compressions.

Original
Administration officials traveling with President Bush in Costa Rica interpreted Mr. Ortega's wavering as a sign that he isn't responding to the military attacks so much as he is searching for ways to strengthen his hand prior to the elections.
Baseline
Administration officials interpreted Mr. Ortega's wavering.
BT-2-Step
Administration officials interpreted Mr. Ortega's wavering as a sign that he isn't responding to the military attacks so much as he is searching for ways.
Spade
Administration officials traveling with President Bush in Costa Rica interpreted Mr. Ortega's wavering as a sign.
Human
Administration officials interpreted Mr. Ortega's wavering as a sign that he is searching for ways to strengthen his hand prior to the elections.

Table 3: Example compressions

Compression	AvgLen	Rating
Baseline	9.70	1.93
BT-2-Step	22.06	3.21
Spade	19.09	3.10
Humans	20.07	3.83

Table 4: Mean ratings for automatic compressions

Procedure and Subjects We obtained compression ratings during an elicitation study completed by 45 unpaid volunteers, all native speaker of English. The study was conducted remotely over the Internet. Participants first saw a set of instructions that explained the task, and defined sentence compression using multiple examples. The materials consisted of the original sentences together with their compressed versions. They were randomised in lists following a Latin square design ensuring that no two compressions in a list were generated from the same sentence. As in Knight and Marcu's (2003) study, participants were asked to use a five point scale to rate the systems' compressions (taking into account the felicity of the compression as well as its grammaticality); they were told that all outputs were generated automatically. Examples of the compressions our participants saw are given in Table 3.

Results We carried out an Analysis of Variance (ANOVA) to examine the effect of different types of compressions (Baseline, BT-2-Step, Spade, and Human). Statistical tests were done using the mean of the ratings shown in Table 4. The ANOVA revealed a reliable effect of compression type by subjects ($F_1(3,90) = 149.50$, $p < 0.001$) and by items ($F_2(3;117) = 40.23$, $p < 0.001$). Post-hoc Tukey tests indicated that human compressions are perceived as significantly better than the compressions produced by the baseline, BT-2-Step, and Spade ($\alpha = 0.01$). The discourse chunker and Spade are significantly better than the baseline ($\alpha = 0.01$). The Tukey test revealed no statistically significant difference between these two algorithms ($\alpha = 0.01$). To summarise, both BoosTexter and Spade perform closer to human performance than the baseline; yet, humans perform significantly better than our compression algorithms.

5 Conclusions

In this paper we proposed discourse chunking as an alternative to full-scale parsing. Central in our approach is the use of low-level syntactic and grammatical information which we argue holds promise for the development of discourse processing models across languages and domains. We showed that a knowledge-lean feature space achieves good performance both on segmentation and span labelling. Furthermore, we assessed the application potential of our chunker and showed that it can be successfully employed to generate sentence compressions, thus confirming one of RST's main claims regarding the nuclearity of discourse spans (at least on the sentence-level).

An important future direction lies in extending our model to the document-level and the assignment of rhetorical relations, thus going beyond the basic nucleus-satellite distinction. Our results indicate that a modular approach to discourse processing (i.e., treating segmentation as separate from labelling) could increase performance. In the future, we plan to investigate how to combine our chunker with models like Spade for improved prediction on both local and global levels.

Acknowledgments

The authors acknowledge the support of EPSRC (Sporleder, grant GR/R40036/01; Lapata, grant GR/T04540/01). Thanks to Amit Dubey, Ben Hutchinson, Alex Lascarides, Simone Teufel, and three anonymous reviewers for helpful comments and suggestions.

References

L. Carlson, D. Marcu, M. E. Okurowski. 2002. RST Discourse Treebank. Linguistic Data Consortium, 2002.

J. Chai, R. Jing. 2004. Discourse structure for context question answering. In *Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*, 23–30.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st NAACL*, 132–139.

B. Endres-Niggemeyer. 1998. *Summarising Information*. Springer, Berlin.

C. Fellbaum, ed. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

J. R. Hobbs, M. Stickel, D. Appelt, P. Martin. 1993. Interpretation as abduction. *Journal of Artificial Intelligence*, 63(1–2):69–142.

K. Humphreys, R. Gaizauskas, S. Azzam. 1997. Event coreference for information extraction. In *Proceedings of the ACL Workshop on Operational Factors in Practical Robust Anaphora Resolution for Unrestricted Texts*, 75–81.

H. Jing, K. McKeown. 2000. Cut and paste summarization. In *Proceedings of the 1st NAACL*, 178–185.

K. Knight, D. Marcu. 2003. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

A. Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

J. Lafferty, A. McCallum, F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*, 282–289.

V. J. Leffa. 1998. Clause processing in complex sentences. In *Proceedings of the 1st LREC*, 937–943.

W. C. Mann, S. A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, ISI, Los Angeles, CA, 1987.

D. Marcu. 1998. To build text summaries of high quality, nuclearity is not sufficient. In *Working Notes of the AAAI-98 Spring Symposium on Intelligent Text Summarization*, 1–8.

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

A. Mikheev. 1997. The LTG part of speech tagger. Technical report, University of Edinburgh, 1997.

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

L. A. Ramshaw, M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL Workshop on Very Large Corpora*, 82–94.

S. Riezler, T. H. King, R. Crouch, A. Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of HLT/NAACL 2003*, 118–125.

R. E. Schapire, Y. Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

R. Soricut, D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of HLT/NAACL 2003*.

S. Teufel, M. Moens. 2002. Summarizing scientific articles – experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–446.

E. F. Tjong Kim Sang. 2000. Text chunking by system combination. In *Proceedings of CoNLL-00*, 151–153.

V. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience, New York.

J. Veenstra. 1998. Fast NP chunking using memory-based learning techniques. In *Proceedings of BENELEARN*, 71–79.

A Comparative Study on Language Model Adaptation Techniques Using New Evaluation Metrics

Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond WA 98052 USA
hisamis@microsoft.com

Jianfeng Gao
Microsoft Research Asia
49 Zhichun Road, Haidian District
Beijing 100080 China
jfgao@microsoft.com

Abstract

This paper presents comparative experimental results on four techniques of language model adaptation, including a maximum *a posteriori* (MAP) method and three discriminative training methods, the boosting algorithm, the average perceptron and the minimum sample risk method, on the task of Japanese Kana-Kanji conversion. We evaluate these techniques beyond simply using the character error rate (CER): the CER results are interpreted using a metric of domain similarity between background and adaptation domains, and are further evaluated by correlating them with a novel metric for measuring the side effects of adapted models. Using these metrics, we show that the discriminative methods are superior to a MAP-based method not only in terms of achieving larger CER reduction, but also of being more robust against the similarity of background and adaptation domains, and achieve larger CER reduction with fewer side effects.

1 Introduction

Language model (LM) adaptation attempts to adjust the parameters of a LM so that it performs well on a particular (sub-)domain of data. Currently, most LMs are based on the Markov assumption that the prediction of a word depends only on the preceding *n*–1 words, but such *n*-gram statistics are known to be extremely susceptible to the characteristics of training samples. This is true even when the data sources are supposedly similar: for example, Rosenfeld (1996) showed that perplexity doubled when a LM trained on the Wall Street Journal (1987-1989) was tested on the AP newswire stories of the same period. This observation, coupled with the fact that training data is available in large quantities only in selected domains, facilitates the need for LM adaptation.

There have been two formulations of the LM adaptation problem. One is the *within-domain adaptation*, in which adapted LMs are created for different topics in a single domain (e.g., Seymore and Rosenfeld, 1997; Clarkson and Robinson, 1997; Chen et al., 1998). In these studies, a domain is defined as a body of text originating from a single source, and the main goal of LM adaptation is to fine-tune the model parameters so as to improve the LM performance on a specific sub-domain (or topic) using the training data at hand.

The other formulation, which is the focus of the current study, is to adapt a LM to a novel domain, for which only a very small amount of training data is available. This is referred to as *cross-domain adaptation*. Following Bellegarda (2001), we call the domain used to train the original model the *background* domain, and the novel domain with a small amount of training data as the *adaptation* domain. Two major approaches to cross-domain adaptation have been investigated: maximum *a posteriori* (MAP) estimation and discriminative training methods. In MAP estimation methods, adaptation data is used to adjust the parameters of the background model so as to maximize the likelihood of the adaptation data. Count merging and linear interpolation of models are the two MAP estimation methods investigated in speech recognition experiments (Iyer et al., 1997; Bacchiani and Roark, 2003), with count merging reported to slightly outperform linear interpolation.

Discriminative approaches to LM adaptation, on the other hand, aim at using the adaptation data to directly minimize the errors on the adaptation data made by the background model. These techniques have been applied successfully to the task of language modeling in non-adaptation (Roark et al.,

2004) as well as adaptation (Bacchiani et al., 2004) scenarios.

In this paper, we present comparative experimental results on four language model adaptation techniques and evaluate them from various angles, attempting to elucidate the characteristics of these models. The four models we compare are a maximum *a posteriori* (MAP) method and three discriminative training methods, namely the boosting algorithm (Collins, 2000), the average perceptron (Collins, 2002) and the minimum sample risk method (Gao et al., 2005). Our evaluation of these techniques is unique in that we go beyond simply comparing them in terms of character error rate (CER): we use a metric of distributional similarity to measure the distance between background and adaptation domains, and attempt to correlate it with the CER of each adaptation method. We also propose a novel metric for measuring the side effects of adapted models using the notion of *backward compatibility*, which is very important from a software deployment perspective.

Our experiments are conducted in the setting of Japanese Kana-Kanji conversion, as we believe this task is excellently suited for evaluating LMs. We begin with the description of this task in the following section.

2 Language Modeling in the Task of IME

This paper studies language modeling in the context of Asian language (e.g., Chinese or Japanese) text input. The standard method for doing this is that the users first input the phonetic strings, which are then converted into the appropriate word string by software. The task of automatic conversion has been the subject of language modeling research in the context of *Pinyin-to-Character conversion* in Chinese (Gao et al., 2002a) and *Kana-Kanji conversion* in Japanese (Gao et al., 2002b). In this paper, we call the task *IME* (Input Method Editor), based on the name of the commonly used Windows-based application.

The performance of IME is typically measured by 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. Current IME systems exhibit about 5-15% CER on real-world data in a wide variety of domains.

In many ways, IME is a similar task to speech recognition. The most obvious similarity is that IME can also be viewed as a Bayesian decision problem: let A be the input phonetic string (which corresponds to the acoustic signal in speech); the task of IME is to choose the most likely word string W^* among those candidates that could have been converted from A:

$$W^* = \underset{W\in\mathbf{GEN}(A)}{\arg\max} P(W \mid A) = \underset{W\in\mathbf{GEN}(A)}{\arg\max} P(W)P(A \mid W) \qquad (1)$$

where $\mathbf{GEN}(A)$ denotes the candidate set given A.

Unlike speech recognition, however, there is no acoustic ambiguity in IME, because the phonetic string is provided directly by users. Moreover, we can assume a unique mapping from W to A in IME, i.e., $P(A|W) = 1$. So the decision of Equation (1) depends solely on $P(W)$, which makes IME ideal for testing language modeling techniques. Another advantage of using IME for language modeling research is that it is relatively easy to convert W to A, which facilitates the creation of training data for discriminative learning, as described later.

From the perspective of LM adaptation, IME faces the same problem speech recognition faces: the quality of the model depends heavily on the similarity of the training and test data. This poses a serious challenge to IME, as it is currently the most widely used method of inputting Chinese or Japanese characters, used by millions of users for inputting text of *any* domain. LM adaptation in IME is therefore an imminent requirement for improving user experience, not only as we build static domain-specific LMs, but also in making online user adaptation possible in the future.

3 Discriminative Algorithms for LM Adaptation

This section describes three discriminative training methods we used in this study. For a detailed description of each algorithm, readers are referred to Collins (2000) for the boosting algorithm, Collins (2002) for perceptron learning, and Gao et al. (2005) for the minimum sample risk method.

3.1 Definition

The following set-up, adapted from Collins (2002), was used for all three discriminative training methods:

- Training data is a set of input-output pairs. In the task of IME, we have training samples $\{A_i, W_i^R\}$, for $i = 1\ldots M$, where each A_i is an input phonetic string and each W_i^R is the reference transcript of A_i.
- 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 $\mathbf{f}(W) = \{f_0(W), f_1(W), \ldots, f_D(W)\}$. The feature $f_0(W)$ is called the *base model* feature, and is defined as the log probability that the word trigram model assigns to W. The features $f_d(W)$ for $d = 1\ldots D$ are defined as the word n-gram counts (n = 1 and 2 in our experiments) in W.
- The parameters of the model form a vector of $D + 1$ dimensions, one for each feature function, $\boldsymbol{\lambda}= \{\lambda_0, \lambda_1, \ldots, \lambda_D\}$. The likelihood score of a word string W can then be written as

$$Score(W,\boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d=0}^{D}\lambda_d f_d(W)\,. \tag{2}$$

Given a model $\boldsymbol{\lambda}$ and an input A, the decision rule of Equation (1) can then be rewritten as

$$W^*(A,\boldsymbol{\lambda}) = \arg\max_{W\in\mathbf{GEN}(A)} Score(W,\boldsymbol{\lambda}). \tag{3}$$

We can obtain the number of conversion errors in W by comparing it with the reference transcript W^R using an error function $\mathrm{Er}(W^R,W)$, which is an edit distance in our case. We call the sum of error counts over the training set the *sample risk* (SR). Discriminative training methods strive to optimize the parameters of a model by minimizing SR, as in Equation (4).

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} SR(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\lambda}} \sum_{i=1\ldots M}\mathrm{Er}(W_i^R, W_i(A_i,\boldsymbol{\lambda})) \tag{4}$$

However, (4) cannot be optimized directly by regular gradient-based procedures as it is a piecewise constant function of $\boldsymbol{\lambda}$ and its gradient is undefined. The discriminative training methods described below differ in how they achieve the optimization: the boosting and perceptron algorithms approximate SR by loss functions that are suitable for optimization; the minimum sample risk method, on the other hand, uses a simple heuristic training procedure to minimize SR directly without resorting to an approximated loss function.

3.2 The boosting algorithm

The boosting algorithm we used is based on Collins (2000). Instead of measuring the number of conversion errors directly, it uses a loss function that measures the number of ranking errors, i.e., cases where an incorrect candidate W receives a higher score than the correct conversion W^R. The *margin* of the pair (W^R, W) with respect to the model λ is given by

$$M(W^R, W) = Score(W^R,\boldsymbol{\lambda}) - Score(W,\boldsymbol{\lambda}) \tag{5}$$

The loss function is then defined as

$$\mathrm{RLoss}(\boldsymbol{\lambda}) = \sum_{i=1\ldots M}\sum_{W_i\in\mathbf{GEN}(A_i)} I[M(W_i^R, W_i)] \tag{6}$$

where $I[\pi] = 1$ if $\pi \le 0$, and 0 otherwise. Note that RLoss takes into account all candidates in $\mathbf{GEN}(A)$.

Since optimizing (6) is NP-complete, the boosting algorithm optimizes its upper bound:

$$\mathrm{ExpLoss}(\boldsymbol{\lambda}) = \sum_{i=1\ldots M}\sum_{W_i\in\mathbf{GEN}(A_i)} \exp(-M(W_i^R, W_i)) \tag{7}$$

Figure 1 summarizes the boosting algorithm we used. After initialization, Step 2 and 3 are repeated N times; at each iteration, a feature is chosen and its weight is updated. We used the following update for the dth feature f_d:

$$\delta_d = \frac{1}{2}\log\frac{C_d^+ + \varepsilon Z}{C_d^- + \varepsilon Z} \tag{8}$$

where C_d^+ is a value increasing exponentially with the sum of margins of (W^R, W) pairs over the set where f_d is seen in W^R but not in W; C_d^- is the value related to the sum of margins over the set where f_d 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.

```
1  Set $\lambda_0 = 1$ and $\lambda_d = 0$ for $d=1\ldots D$
2  Select a feature $f_d$ which has largest estimated im-
   pact on reducing ExpLoss of Equation (7)
3  Update $\lambda_d$ by Equation (8), and return to Step 2
```

Figure 1: The boosting algorithm

3.3 The perceptron algorithm

The perceptron algorithm can be viewed as a form of incremental training procedure that optimizes a *minimum square error* (MSE) loss function, which is an approximation of SR (Mitchell, 1997). As shown in Figure 2**,** it starts with an initial parameter setting and updates it for each training sample. We used the average perceptron algorithm of Collins (2002) in our experiments, a variation that has been proven to be more effective than the standard algorithm shown in Figure 2. Let $\lambda_d^{t,i}$ be the

```
1 Set $\lambda_0$ = 1 and $\lambda_d$ = 0 for $d$=1…$D$
2 For $t$ = 1…$T$ ($T$ = the total number of iterations)
3   For each training sample ($A_i$, $W_i^R$), $i$ = 1…$M$
4     Choose the best candidate $W_i$ from GEN($A_i$)
      according to Equation (3)
5     For each $\lambda_d$ ($\eta$ = size of learning step)
6       $\lambda_d = \lambda_d + \eta(f_d(W_i^R) - f_d(W_i))$
```

Figure 2: The perceptron algorithm

value for the *d*th parameter after the *i*th training sample has been processed in pass *t* over the training data. The average parameters are defined as

$$(\lambda_d)_{avg} = (\sum_{t=1}^{T}\sum_{i=1}^{M}\lambda_d^{t,i})/(T \cdot M). \tag{9}$$

3.4 The minimum sample risk method

The minimum sample risk (MSR, Gao et al., 2005) training algorithm is motivated by analogy with the feature selection procedure for the boosting algorithm (Freund et al., 1998). It is a greedy procedure for selecting a small subset of the features that have the largest contribution in reducing SR in a sequential manner. Conceptually, MSR operates like any multidimensional function optimization approach: a direction (i.e., feature) is selected and SR is minimized along that direction using a *line search*, i.e., adjusting the parameter of the selected feature while keeping all other parameters fixed. This is repeated until SR stops decreasing.

Regular numerical line search algorithms cannot be applied directly because, as described above, the value of a feature parameter versus SR is not smooth and there are many local minima. MSR thus adopts the method proposed by Och (2003). Let **GEN**(A) be the set of n-best candidate word strings that could be converted from A. By adjusting λ_d for a selected feature f_d, we can find a set of intervals for λ_d within which a particular candidate word string is selected. We can compute *Er*(.) for the candidate and use it as the *Er*(.) value for the corresponding interval. As a result, we obtain an ordered sequence of *Er*(.) values and a corresponding sequence of λ intervals for each training sample. By summing *Er*(.) values over all training samples, we obtain a global sequence of SR and the corresponding global sequence of λ_d intervals. We can then find the optimal λ_d as well as its corresponding SR by traversing the sequence.

Figure 3 summarizes the MSR algorithm. See Gao et al. (2005) for a complete description of the MSR implementation and the empirical justification for its performance.

```
1 Set $\lambda_0$ = 1 and $\lambda_d$ = 0 for $d$=1…$D$
2 Rank all features by its expected impact on reduc-
  ing SR and select the top $N$ features
3 For each $n$ = 1…$N$
4   Update the parameter of $f$ using line search
```

Figure 3: The MSR algorithm

4 Experimental Results

4.1 Data

The data used in our experiments come from five distinct sources of text. A 36-million-word *Nikkei* newspaper corpus was used as the background domain. We used four adaptation domains: *Yomiuri* (newspaper corpus), *TuneUp* (balanced corpus containing newspaper and other sources of text), *Encarta* (encyclopedia) and *Shincho* (collection of novels). The characteristics of these domains are measured using the information theoretic notion of cross entropy, which is described in the next subsection.

For the experiment of LM adaptation, we used the training data consisting of 8,000 sentences and test data of 5,000 sentences from each adaptation domain. Another 5,000-sentence subset was used as held-out data for each domain, which was used to determine the values of tunable parameters. All the corpora used in our experiments are pre-segmented into words using a baseline lexicon consisting of 167,107 entries.

4.2 Computation of domain characteristics

Yuan et al. (2005) introduces two notions of domain characteristics: a within-domain notion of *diversity*, and a cross-domain concept of *similarity*. Diversity is measured by the entropy of the corpus and indicates the inherent variability within the domain. Similarity, on the other hand, is intended to capture the difficulty of a given adaptation task, and is measured by the cross entropy.

For the computation of these metrics, we extracted 1 million words from the training data of each domain respectively, and created a lexicon consisting of the words in our baseline lexicon plus all words in the corpora used for this experiment (resulting in 216,565 entries) to avoid the effect of out-of-vocabulary items. Given two domains *A* and

	N	Y	T	E	S
Nikkei	**3.94**	7.46	7.65	9.81	10.10
Yomiuri		**4.09**	7.82	8.96	9.29
TuneUp			**4.41**	8.82	8.56
Encarta				**4.40**	9.20
Shincho					**4.61**

Table 1: Cross entropy

Domain	Base	LI	MSR	Boost	Percep
Yomiuri	3.70	3.69	2.89	2.88	**2.85**
TuneUp	5.81	5.70	5.48	**5.47**	**5.47**
Encarta	10.24	8.64	8.39	8.54	**8.34**
Shincho	12.18	11.47	**11.05**	11.09	11.20

Table 2: CER results (%) (Base=baseline model; LI=linear interpolation)

B, we then trained a word trigram model for each domain *B*, and used the resulting model in computing the cross entropy of domain *A*. For simplicity, we denote this as *H*(*A*,*B*).

Table 1 summarizes our corpora along this dimension. Note that the cross entropy is not symmetric, i.e., H(*A*,*B*) is not necessarily the same as H(*B*,*A*), so we only present the *average cross entropy* in Table 1. We can observe that Yomiuri and TuneUp are much more *similar* to the background Nikkei corpus than Encarta and Shincho.

H(*A*,*A*) along the diagonal of Table 1 (in boldface) is the entropy of the corpus, indicating the corpus *diversity*. This quantity indeed reflects the in-domain variability of text: newspaper and encyclopedia articles are highly edited text, following style guidelines and often with repetitious content. In contrast, Shincho is a collection of novels, on which no style or content restriction is imposed. We use these metrics in the interpretation of CER results in Section 5.

4.3 Results of LM adaptation

The discriminative training procedure was carried out as follows: for each input phonetic string *A* in the adaptation training set, we produced a word lattice using the baseline trigram models described in Gao et al. (2002b). We kept the top 20 hypotheses from this lattice as the candidate conversion set **GEN**(*A*). The lowest CER hypothesis in the lattice rather than the reference transcript was used as W^R. We used unigram and bigram features that occurred more than once in the training set.

We compared the performance of discriminative methods against a MAP estimation method as the baseline, in this case the linear interpolation method. Specifically, we created a word trigram model using the adaptation data for each domain, which was then linearly interpolated at the word level with the baseline model. The probability according to the combined model is given by

$$p(w_i \mid h) = \lambda P_B(w_i \mid h) + (1-\lambda) P_A(w_i \mid h) ,$$

where P_B is the probability of the background model, P_A the probability of the adaptation model, and the history *h* corresponds to two preceding words. λ was tuned using the held-out data.

In evaluating both MAP estimation and discriminative models, we used an *N*-best rescoring approach. That is, we created *N* best hypotheses using the baseline trigram model (*N*=100 in our experiments) for each sentence in the test data, and used adapted models to rescore the *N*-best list. The oracle CERs (i.e., the minimal possible CER given the available hypotheses) ranged from 1.45% to 5.09% depending on the adaptation domain.

The results of the experiments are shown in Table 2. We can make some observations from the table. First, all discriminative methods significantly outperform the linear interpolation (statistically significant according to the *t*-test at $p < 0.01$). In contrast, the differences among three discriminative methods are very subtle and most of them are not statistically significant. Secondly, the CER results correlate well with the metric of domain similarity in Table 1 (*r*=0.94 using the Pearson product moment correlation coefficient). This is consistent with our intuition that the closer the adaptation domain is to the background domain, the easier the adaptation task.

Regarding the similarity of the adaptation domain to the background, we also observe that the CER reduction of the linear interpolation model is particularly limited when the adaptation domain is similar to the background domain: the CER reduction of the linear interpolation model for Yomiuri and TuneUp over the baseline is 0% and 1.89% respectively, in contrast to ~22% and ~5.8% improvements achieved by the discriminative models. The discriminative methods are therefore more robust against the similarity of the adaptation and background data than the linear interpolation.

Our results differ from Bacchiani et al. (2004) in that in our system, the perceptron algorithm alone achieved better results than MAP estimation. However, the difference may only be apparent, given different experimental settings for the two studies. We used the *N*-best reranking approach with the same *N*-best list for both MAP estimation and discriminative training, while in Bacchiani et al. (2004), two different lattices were used: the perceptron model was applied to rerank the lattice created by the background model, while the MAP adaptation model was used to produce the lattice itself. The fact that the combination of these models (i.e., first use the MAP estimation to create hypotheses and then use the perceptron algorithm to rerank them) produced the best results indicates that given a candidate lattice, the perceptron algorithm is effective in candidate reranking, thus making our results compatible with theirs.

5 Discussion

The results in Section 4 demonstrate that discriminative training methods for adaptation are overall superior to MAP adaptation methods. In this section, we show additional advantages of discriminative methods beyond simple CER improvements.

5.1 Using metrics for side effects

In the actual deployment of LM adaptation, one issue that bears particular importance is the number of side effects that are introduced by an adapted model. For example, consider an adapted model which achieves 10% CER improvements over the baseline. Such a model can be obtained by improving 10%, or by improving 20% *and* by introducing 10% of new errors. Clearly, the former model is preferred, particularly if the models before and after adaptation are both to be exposed to users. This concept is more widely acknowledged within the software industry as *backward compatibility* – a requirement that an updated version of software supports all features of its earlier versions. In IME, it means that all phonetic strings that can be converted correctly by the earlier versions of the system should also be converted correctly by the new system as much as possible. Users are typically more intolerant to seeing errors on the strings that used to be converted correctly than seeing errors that also existed in the previous version. Therefore, it is crucial that when we adapt to a new domain, we do so by introducing the smallest number of side effects, particularly in the case of an incremental adaptation to the domain of a particular *user*, i.e., to building a model with incremental learning capabilities.

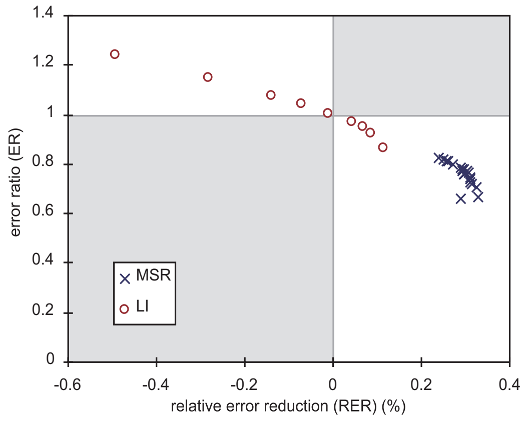


Figure 4: RER/ER plot for MSR and LI models for TuneUp domain

5.2 Error ratio

In order to measure side effects, we introduce the notion of *error ratio* (ER), which is defined as

$$ER = \frac{|E_A|}{|E_B|},$$

where $|E_A|$ is the number of errors found *only* in the new (adaptation) model, and $|E_B|$ the number of errors corrected by the new model. Intuitively, this quantity captures the *cost* of improvement in the adaptation model, corresponding to the number of newly introduced errors per each improvement. The smaller the ratio is, the better the model is at the same CER: ER=0 if the adapted model introduces no new errors, ER<1 if the adapted model makes CER improvements, ER=1 if the CER improvement is zero (i.e., the adapted model makes as many new mistakes as it corrects old mistakes), and ER>1 when the adapted model has worse CER performance than the baseline model.

Given the notion of CER and ER, a model can be plotted on a graph as in Figure 4: the *relative error reduction* (RER, i.e., the CER difference between the background and adapted models) is plotted along the x-axis, and ER along the y-axis. Figure 4 plots the models obtained after various numbers of iterations for MSR training and at various interpolation weights for linear interpolation for the TuneUp domain. The points in the upper-left quadrant, ER>1 and RER<0, are the models that performed worse than the baseline model (some of the interpolated models fall into this category); the shaded areas (upper-right and lower-left quadrants) are by definition empty. The lower-right quadrant is the area of interest to us, as they

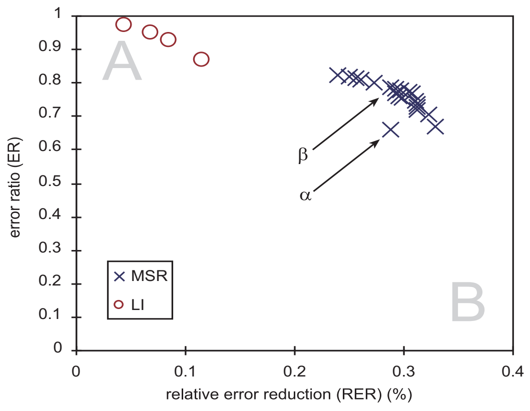


Figure 5: RER/ER plot for the models with ER<1 and RER>0 for TuneUp domain. See Figure 8 for the description of *α* and *β*

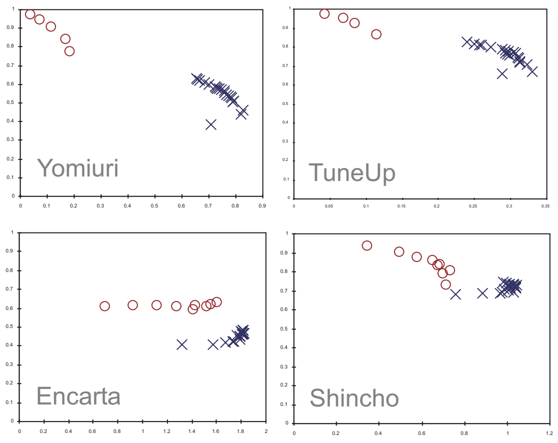


Figure 6: RER/ER plot for all four domains
x-axes: RER (%); y-axes: ER
○: linear interpolation models; ×:MSR models

represent the models that led to CER improvements; we will focus only on this area now in Figure 5.

In this figure, a model is considered to have fewer side effects when the ER is smaller at the same RER (i.e., smaller value of *y* for a fixed value of *x*), or when the RER is larger at the same ER (i.e., larger value of *x* at the fixed *y*). That is, the closer a model is plotted to the corner B of the graph, the better the model is; the closer it is to the corner A, the worse the model is.

5.3 Model comparison using RER/ER

From Figure 5, we can clearly see that MSR models have better RER/ER-performance than linear interpolation models, as they are plotted closer to the corner B. Figure 6 displays the same plot for all four domains: the same trend is clear from all graphs. We can therefore conclude that a discriminative method (in this case MSR) is superior to linear interpolation not only in terms of CER reduction, but also of having fewer side effects. This desirable result is attributed to the nature of discriminative training, which works specifically to adjust feature weights so as to minimize error.

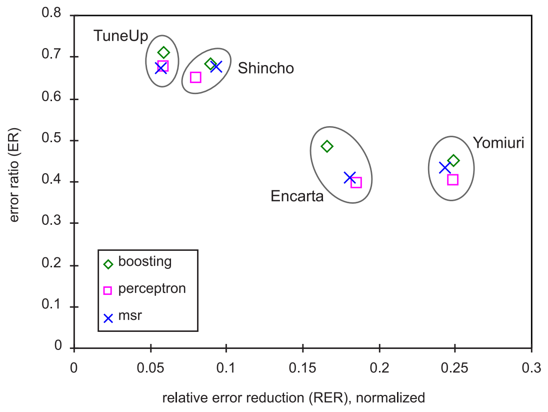


Figure 7: RER/ER plot for MSR, boosting and perceptron models (X-axis is normalized to represent relative error *rate* reduction)

Figure 7 compares the three discriminative models with respect to RER/ER by plotting the best models (i.e., models used to produce the results in Table 1) for each algorithm. We can see that even though the boosting and perceptron algorithms have the same CER for Yomiuri and TuneUp from Table 2, the perceptron is better in terms of ER; this may be due to the use of exponential loss function in the boosting algorithm which is less robust against noisy data (Hastie et al., 2001). We also observe that Yomiuri and Encarta do better in terms of side effects than TuneUp and Shincho for all algorithms, which can be explained by corpus diversity, as the former set is less stylistically diverse and thus more consistent within the domain.

5.4 Overfitting and side effects

The RER/ER graph also casts the problem of *overfitting* in an interesting perspective. Figure 8 is derived from running MSR on the TuneUp test corpus, which depicts a typical case of overfitting: the CER drops in the beginning, but after a certain number of iterations, it goes up again. The models indicated by *α* and *β* in the graph are of the same CER, and as such, these models are equivalent. When plotted on the RER/ER graph in Figure 5,

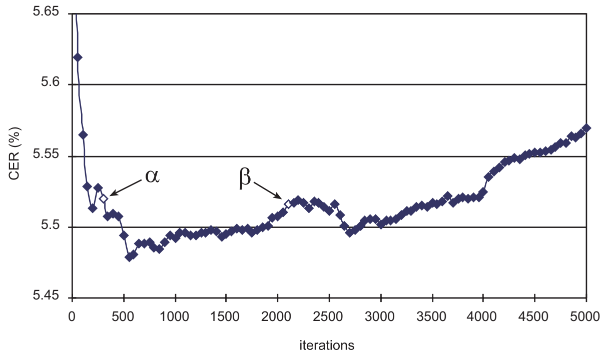


Figure 8: MSR test error curve for TuneUp

however, it is clear that the overfit model β has the worse ER than the non-overfit counterpart α. In other words, models α and β have the same CER, but they are not equivalent: model β is not only worse in light of containing more features, but also in terms of causing more side effects.

6 Conclusion and Future Work

We have presented a comparison of three discriminative learning approaches with a MAP estimation method in the task of LM adaptation for IME. We have shown that all discriminative models are significantly better than the linear interpolation method, in that they achieve larger CER reduction with fewer side effects across different domains.

One direction of future research is to apply this technique to an incremental learning scenario, i.e., to incrementally build models using incoming data for adaptation, taking all previously available data as background corpus. The new metric for backward compatibility we proposed in the paper will play a particularly important role in such a scenario.

Acknowledgements

We would like to thank Kevin Duh, Gary Kacmarcik, Eric Ringger, Yoshiharu Sato and Wei Yuan for their help at various stages of this research.

References

Bacchiani, M. and B. Roark. 2003. Unsupervised Language Model Adaptation. *Proceedings of ICASSP*, pp.224-227.

Bacchiani, M., B. Roark and M. Saraclar. 2004. Language Model Adaptation with MAP Estimation and the Perceptron Algorithm. *Proceedings of HLT-NAACL*, pp.21-24.

Bellegarda, J.R. 2001. An Overview of Statistical Language Model Adaptation. *ITRW on Adaptation Methods for Speech Recognition*, pp. 165-174.

Chen, S.F., K. Seymore and R. Rosenfeld. 1998. Topic Adaptation for Language Modeling Using Unnormalized Exponential Models. *Proceedings of ICASSP*.

Clarkson, P.R., and A.J. Robinson. 1997. Language Model Adaptation Using Mixtures and an Exponentially Decaying Cache. *Proceedings of ICASSP*.

Collins, M. 2000. Discriminative Reranking for Natural Language Parsing. *ICML 2000*.

Collins, M. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithm. *Proceedings of EMNLP*, pp.1-8.

Freund, Y., R. Iyer, R.E. Shapire and Y. Singer. 1998 An Efficient Boosting Algorithm for Combining Preferences. *ICML'98*.

Gao, J., J. Goodman, M. Li and K.-F. Lee. 2002a. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, 1-1: 3-33.

Gao, J, H. Suzuki and Y. Wen. 2002b. Using Headword Dependency and Predictive Clustering for Language Modeling. *Proceedings of EMNLP*: 248-256.

Gao, J., H. Yu, P. Xu and W. Yuan. 2005. Minimum Sample Risk Methods for Language Modeling. *Proceedings of EMNLP 2005*.

Hastie, T., R. Tibshirani and J. Friedman. 2001. *The Elements of Statistical Learning*. Springer-Verlag, New York.

Iyer, R., M. Ostendorf and H. Gish. 1997. Using Out-of-Domain Data to Improve In-Domain Language Models. *IEEE Signal Processing Letters*, 4-8: 221-223.

Mitchell, Tom M. 1997. *Machine learning*. The McGraw-Hill Companies, Inc.

Och, F. J. 2003. Minimum Error Rate Training in Statistical Machine Translation. *Proceedings of ACL*: 160-167.

Roark, B., M. Saraclar and M. Collins. 2004. Corrective Language Modeling for Large Vocabulary ASR with the Perceptron Algorithm. *Proceedings of ICASSP*: 749-752.

Rosenfeld, R. 1996. A Maximum Entropy Approach to Adaptive Statistical Language Modeling. *Computer, Speech and Language*, 10: 187-228.

Seymore, K. and R. Rosenfeld. 1997. Using Story Topics For Language Model Adaptation. *Proceedings of Eurospeech '97*.

Yuan, W., J. Gao and H. Suzuki. 2005. An Empirical Study on Language Model Adaptation Using a Metric of Domain Similarity. *Proceedings of IJCNLP 05*.

PP-attachment disambiguation using large context

Marian Olteanu and Dan Moldovan
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75080
marian@hlt.utdallas.edu
moldovan@utdallas.edu

Abstract

Prepositional Phrase-attachment is a common source of ambiguity in natural language. The previous approaches use limited information to solve the ambiguity – four lexical heads – although humans disambiguate much better when the full sentence is available. We propose to solve the PP-attachment ambiguity with a Support Vector Machines learning model that uses complex syntactic and semantic features as well as unsupervised information obtained from the World Wide Web. The system was tested on several datasets obtaining an accuracy of 93.62% on a Penn Treebank-II dataset; 91.79% on a FrameNet dataset when no manually-annotated semantic information is provided and 92.85% when semantic information is provided.

1 Problem description

1.1 PP-attachment ambiguity problem

Prepositional Phrase-attachment is a source of ambiguity in natural language that generates a significant number of errors in syntactic parsing. For example the sentence "I saw yesterday the man in the park with a telescope" has 5 different semantic interpretations based on the way the prepositional phrases "in the park" and "with the telescope" are attached: *I saw yesterday [the man [in the park [with a telescope]]]*; *I saw yesterday [the man [in the park] [with a telescope]]*; *I saw yesterday [the man [in the park]] [with a telescope]*; *I saw yesterday [the man] [in the park [with a telescope]]* and *I saw yesterday [the man] [in the park] [with a telescope].*

The problem can be viewed as a decision of attaching a prepositional phrase (PP) to one of the preceding head nouns or verbs. The ambiguity expressed by the number of potential parse trees generated by Context-Free Grammars increases exponentially with the number of PPs. For a PP that follows the object of a verb there are 2 parse trees, for a chain of 2, 3, 4 and 5 PPs there are respectively 5, 14, 42 and 132 parse trees. Usually the average number of consecutive PPs in a sentence increases linearly with the length of the sentence.

Lexical and syntactic information alone is not sufficient to resolve the PP-attachment problem; often semantic and/or contextual information is necessary. For example, in *"I ate a pizza with anchovies"*, *"with anchovies"* attaches to the noun "pizza", where as in *"I ate a pizza with friends."*, *"with friends"* attaches to the verb "eat" – example found in (McLauchlan, 2001). There are instances of PP-attachment, like the one in *"I saw the car in the picture"* that can be disambiguated only by using contextual discourse information.

Usually, people don't have much trouble in finding the right way to attach PPs. But if one limits the information used for disambiguation of the PP-attachment to include only the verb, the noun representing its object, the preposition and the main noun in the PP, the accuracy for human decision degrades from 93.2% to 88.2% (Ratnaparkhi et al., 1994) on a dataset extracted from Penn Treebank (Marcus et al., 1993).

1.2 Motivation

Syntactic parsing is essential for many natural language applications such as Machine Translation, Question Answering, Information Extraction, Information Retrieval, Automatic Speech Recognition. Since parsing occurs early in the chain of NLP processing steps it has a large impact on the overall system performance.

2 Approach

Our approach to solve the PP-attachment ambiguity is based on a Support Vector Machines learner (Cortes and Vapnik, 1995). The feature set contains complex information extracted automatically from candidate syntax trees generated by parsing (Charniak, 2000), trees that will be improved by more accurate PP-attachment decisions. Some of these features were proven efficient for semantic information labeling (Gildea and Jurafsky, 2002). The feature set also includes unsupervised information obtained from a very large corpus (World Wide Web). Features containing manually annotated semantic information about the verb and about the objects of the verb have also been used. We adopted the standard approach to distinguish between verb and noun attachment; thus the classifier has to choose between two classes: V when the prepositional phrase is attached to the verb and N when the prepositional phrase is attached to the preceding head noun.

3 Data

To be able to extract the required features from a dataset instance, one must identify the verb, the phrase identifying the object of the verb that precedes the prepositional phrase in question ($np1$) which usually is part of the predicate-argument structure of the verb, its head noun, the prepositional phrase ($np2$), its preposition and its head noun (the second most important word in the PP).

We have adopted the notation from (Collins and Brooks, 1995), where v is the verb, n_1 is the head noun of object phrase, p is the preposition and n_2 is the head noun of the prepositional phrase.

Compared to our datasets, Ratnaparkhi's dataset (Ratnaparkhi et al., 1994) contains only the lexical heads v, n_1, p and n_2. Thus, our methodology cannot be applied to Ratnaparkhi's dataset (RRR).

In our experiments we used two datasets:

- **FN** – extracted from FrameNet II 1.1 (Baker et al., 1998)
- **TB2** – extracted from Penn Treebank-II

Table 1 presents the datasets[1]. The creation of the datasets is described in details in (Olteanu, 2004).

4 Features

The experiments described in this paper use a set of discrete (alphanumeric) and continuous (numeric) features. All features are fully deterministic, except the features **count-ratio** and **pp-count** that are based on information provided by an external resource - Google search engine (`http://www.google.com`).

In describing the features, we will use the Penn Treebank-II parse tree associated with the sentence *"The Lorillard spokeswoman said asbestos was used in "very modest amounts" in making paper for the filters in the early 1950s and replaced with a different type of filter in 1956"*.

Table 2 describes the features and the origin of each feature. The preposition is the feature with the most discriminative power, because of preferences of particular prepositions to attach to verbs or nouns. Table 3 shows the distribution of top 10 most frequently used prepositions in the FN and TB2 datasets.

The features were carefully designed so that, when they are extracted from gold parse trees, they don't provide more information useful for disambiguation than when they are automatically generated using a parser. This claim is validated by the experimental results that show a strong correlation between the results on the two datasets – one based on automatically generated parse trees (FN) and one based on gold parse trees (TB2).

Next, we describe in further detail the features presented in Table 2.

v-frame represents the *frame of the verb* – the frame to which the verb belongs, as it is present in FrameNet (manually annotated). We used this feature because the frame of the verb describes very well the semantic behavior of the verb including the predicate-argument structure of the verb, which entails the affinity of the verb for certain prepositions.

[1]The datasets are available at `http://www.utdallas.edu/~mgo031000/ppa/`

	FN	**TB2**
Source	FrameNet annotation samples (British National Corpus)	Penn Treebank-II (WSJ articles)
Instance identification	Semantic-centered (related to Frame Elements)	Syntactic-centered (related to the structure of the parse tree)
Parse trees	Automatically generated (Charniak)	Gold standard
Total size	27,421 instances	60,699 instances
Distribution statistics	70.28% ambiguous verb attachments 2.36:1 v-attch:n-attch	35.71% ambiguous verb attachments 1:1.8 v-attch:n-attch
Training / test sets	90% - 10% – homogenously distributed (one in every 10 instances is selected for the test set)	
Location of PP	Both before and after verb	Only after verb
Other properties	– Partial identification of ambiguous PP-attachment instances in the corpus, derived from manual annotation of FEs (Olteanu, 2004) – Semantic information readily available	

Table 1: The datasets and their characteristics

Feature: description *[origin]*
v-surface: surface form of the verb *[Hindle'93, ...]*
n1-surface: surface form of n_1. May be morphologically processed *[Hindle'93, ...]*
p: the preposition, lower-cased *[Hindle'93, ...]*
n2-surface: surface form of n_2. May be morphologically processed *[Ratnaparkhi'94, Collins'95, ...]*
n1-mp/n1-mpf: morph. processing of n_1 *[Collins'95]*
n2-mp/n2-mpf: morph. processing of n_2 *[Collins'95]*
v-lemma: lemma of the verb *[Collins'95]*
path: path in the candidate parse tree between the verb and *np1* *[Gildea'02]*
subcategorization: subcategorization of the verb *[modified from Pradhan'03]*
v-pos: part-of-speech of the verb
v-voice: voice of the verb
n1-pos: part-of-speech of n_1
n1-lemma: lemma of n_1. May be morphologically processed
n2-pos: part-of-speech of n_2
n2-lemma: lemma of n_2. May be morphologically processed
position: position of *np1* relative to the verb *[new]*
v-frame: frame of the verb *[new in PPA]*
n1-sr: semantic role of *np1* *[new in PPA]*
n1-tr: thematic role of *np1* *[new in PPA]*
n1-preposition: preposition that heads *np1*, if *np1* is a PP *[new]*
n1-parent: label of the parent of *np1* in the candidate parse tree *[new in PPA]*
n1-np-label: label of *np1* in the candidate parse tree *[new in PPA]*
n2-det: determination of *np2* *[new]*
parser-vote: choice of the automatic parser in attaching PP *[new in PPA]*
count-ratio: WWW statistics about verb-attachment vs. noun-attachment for that particular instance *[new]*
pp-count: WWW statistics about co-occurrence of *v* and n_2 *[new]*
n1-p-distance: the distance between n_1 and *p* *[new]*

Table 2: Features

Prep.	**% of FN**	**% v-att FN**	**% of TB2**	**% v-att TB2**
of	13.47%	6.17%	30.14%	2.74%
to	13.27%	80.14%	9.55%	60.49%
in	12.42%	73.64%	16.94%	42.58%
for	6.87%	82.44%	8.95%	39.72%
on	6.21%	75.51%	5.16%	47.73%
with	6.17%	86.30%	3.79%	46.92%
from	5.37%	75.90%	5.76%	52.76%
at	4.09%	76.63%	3.21%	66.02%
as	3.95%	86.51%	2.49%	51.69%
by	3.53%	88.02%	3.27%	68.11%

Table 3: Distribution of the first 10 most-frequent prepositions in the FN and TB2 datasets

n1-sr represents the *semantic role of the object phrase np1* – the label attached to the Frame Element (manual semantic annotation that can be found in FrameNet). This feature was introduced because of the relation between the underlying meaning of *np1* and its semantic role.

n1-tr represents the *thematic role of the object phrase np1* – a coarse-grained role based on the label attached to the Frame Element (manual semantic annotation that can be found in FrameNet). It was introduced to reduce data sparseness for the **n1-sr** feature. The conversion from fine-grained semantic role to coarse-grained semantic role is done automatically using a table that maps a pair of a frame-level semantic role (FE label) and a frame to a thematic role.

subcategorization contains a semi-lexicalized description of the structure of the verb phrase. A subcategorization frame is closely related to the predicate argument structure and to the underlying meaning of the verb. It contains an ordered set of all the phrase labels that are siblings of the verb, plus a marker for the verb. If the child phrase of the verb is a PP, then the label will also contain the preposition (the headword of the PP). This feature is a modified form of the sub-categorization feature described in (Pradhan et al., 2003): the differences in various part-of-speeches for the verb were ignored and the preposition that heads a prepositional phrase is also attached to the label. Therefore, for the sentence "*The stock declined in June by 4%*", the value for this feature is **-PPin-PPby*.

In the TB2 dataset the parse trees are gold standard (contain the expected output value for PP-ambiguity resolution). In the case of a verb attachment, if the selected PP is a child of the selected VP, then by applying the algorithm, the value of the feature will contain the PP label plus the preposition. This clearly is a clue for the learner that the instance is a *verb attachment*. To overcome this problem for datasets based on gold-standard parse trees, when computing the value of the **subcategorization** feature the selected PP will not be used. Figure 1 shows the subcategorization for the phrase *"replaced with a different type of filter in 1956"*.

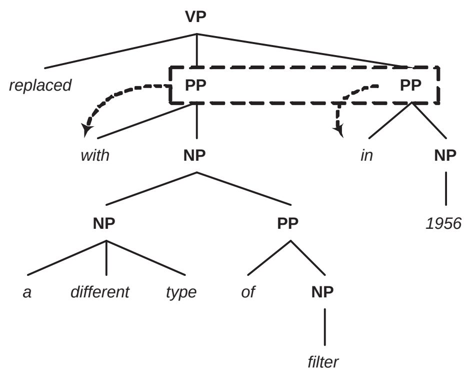


Figure 1: Subcategorization feature: **-PPin-PPby*

path expresses the syntactic relation between the verb v and the object phrase *np1*. Its purpose is to describe the syntactic relation of np1 to the rest of the clause by the syntactic relation of *np1* with the head of the clause – v. We adopted this feature from (Gildea and Jurafsky, 2002). **path** describes the chain of labels in the tree from v to *np1*, including the label of v and *np1*. Ascending movements and descending movements are depicted separately. We used two variants of this feature to determine the optimum version for our problem – one with full POS of the verb and one with POS reduced to "VB". The experiments proved that the second variant provides a better performance. Figure 2 depicts the path between "replaced" and "a different type of filter": VBN↑VP↓PP↓NP or VB↑VP↓PP↓NP.

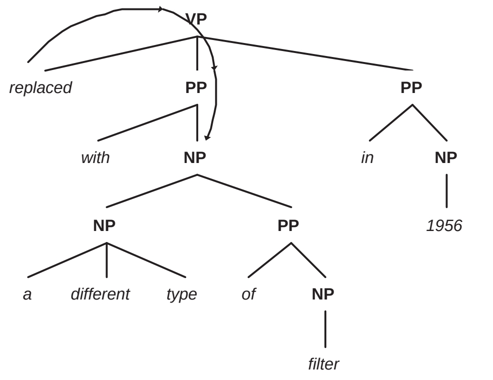


Figure 2: Example of a path feature

position indicates the *position of the n_1-p-n_2 construction relative to the verb*, i.e. whether the prepositional phrase in question lies before the verb or after the verb in the sentence. Position is very important in deciding the type of attachment, considering the totally different distribution of PPs constructions preceding the verb and PPs constructions following the verb.

Morphological processing applied to n_1 and n_2 was inspired by the algorithm described in (Collins and Brooks, 1995). We analyzed the impact of different levels of morphological processing by using two types: partial morphological processing (only numbers and years are converted) – identified by adding **-mp** as a suffix to the name of this feature – and full morphological processing (numbers, years and capitalized names) – identified by adding **-mpf** as a suffix to the name of this feature. The purpose of morphological processing is data sparseness reduction by clustering similar values for this feature.

n1-parent represents the phrase label of the parent of np1 and it cannot be used on gold parse trees (TB2 dataset) because it will provide a clue about the correct attachment type.

n2-det is called the *determination of the prepositional phrase np2*. This novel feature tells if n_2 is preceded in *np2* by a possessive pronoun or by a determiner. This is used to differentiate between *"buy books for children"* (which is probably a noun attachment) and *"buy books for her children"* (which very probably is a verb attachment).

parser-vote feature represents the choice of the parser (Charniak's parser) in the PP-attachment resolution. It cannot be used with gold-standard parse trees because it will provide the right answer.

count-ratio represents the estimated ratio between the frequency of an unambiguous verb attachment construction based on v, p and n_2 and the frequency of a probably unambiguous noun attachment construction based on n_1, p and n_2 in a very large corpus. A very large corpus is required to overcome the data sparseness inherent for complex constructions like those described above.

We chose the World Wide Web as a corpus and Google as a query interface (see (Olteanu, 2004) for details).

Let's consider the estimated frequency of unambiguous verb-attachments and respectively noun-attachments defined as:

$$f_v = \frac{c_{v-p-n2}}{c_v \cdot c_{p-n2}}$$

$$f_n = \frac{c_{n1-p-n2}}{c_{n1} \cdot c_{p-n2}}$$

where:

- c_{v-p-n2} is the number of occurrences of the phrase "v p n_2", "v $p{*}n_2$" (where * symbolizes any word), "*v-lemma p* n_2" or "*v-lemma p * n_2*" in World Wide Web, as reported by Google
- c_v is the number of occurrences of the word "v" or "*v-lemma*" in WWW
- c_{p-n2} is the number of occurrences of the phrase "p n_2" or "$p * n_2$" in WWW
- $c_{n1-p-n2}$ is the number of occurrences of the phrase "n_1 p n_2" or "v $p * n_2$" in WWW
- c_{n1} is the number of occurrences of the word "n_1" in WWW

The value for this feature is:

$$\mathbf{count-ratio} = \log_{10} \frac{f_v}{f_n} = \log_{10} \frac{c_{v-p-n2} \cdot c_{n1}}{c_{n1-p-n2} \cdot c_v}$$

We chose logarithmic values for this feature because experiments showed that logarithmic values provide a higher accuracy than linear values. Also, by experimentation we concluded that value bounding is helpful, and the feature was bounded to values between -3 and 3 on the logarithmic scale, unless specified otherwise in the experiment description.

This feature resembles the approach adopted in (Volk, 2001).

pp-count depicts the estimated count of occurrences in World Wide Web of the prepositional phrases based on p and n_2. The count is estimated by c_{p-n2}. Therefore **pp-count** = $log_{10}(c_{p-n2} + c_{p-*-n2})$.

n1-p-distance depicts the distance (in tokens) between n_1 and p. Let d_{n1-p} be the distance between n_1 and p ($d = 1$ if there is no other token between n_1 and p). Thus **n1-p-distance** = $\log_{10}(1 + \log_{10} d_{n1-p})$.

5 Learning model and procedure

We used in our experiments a Support Vector Machines learner with Radial Basis Function kernel as implemented in the LIBSVM toolkit (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`).

We converted the feature tuples (containing discrete alphanumeric and continuous values) to multi-dimensional vectors using the following procedure:

- Discrete features: assign to each possible value of each feature a dimension in the vector space, and to each feature value in each training or test example put 1 in the dimension corresponding to the feature value and 0 in all other dimensions associated with that feature.
- Continuous features: assign a dimension and put the scaled value in the multi-dimensional vector (all examples in training data will span between 0 and 1 for that particular dimension).

SVM training was preceded by finding the optimal γ and C parameters required for training using 2-fold cross validation, which was found to be superior in model accuracy and training time over higher folds cross-validations (Olteanu, 2004).

The criterion for selecting the best set of features was the accuracy on the cross-validation. Thus, the development of the models was performed entirely on the training set, which acted also as a development set. We later computed the accuracy on the test set on some representative models.

6 Experiments, results and analysis

For each dataset, we conducted experiments to determine an efficient combination of features and the accuracy on test data for the best combination of features. We also run the experimental procedure on the original Ratnaparkhi's dataset in order to compare SVM with other machine learning techniques applied to PP-attachment problem. Table 4 summarizes the experiments performed on all datasets.

Experiment	% on dev / x-val	% on test
FN-basic-flw	86.25	86.44
FN-lex-syn-flw	88.55	89.61
FN-best-no-sem	90.93	91.79
FN-best-sem	91.87	92.85
TB2-basic	85.75	87.47
TB2-best-no-www	92.06	92.81
TB2-best	92.92	93.62
RRR-basic	84.32	84.60
RRR-basic-mpf	84.34	85.14

Table 4: Results

FN-basic-flw uses **v-surface**, **n1-surface**, **p** and **n2-surface** on examples that follow the verb. *FN-lex-syn-flw* uses **v-surface**, **v-pos**, **v-lemma**, **subcategorization**, **path** (full POS), **position**, **n1-preposition**, **n1-surface**, **n1-pos**, **n1-lemma**, **n1-parent**, **p**, **n2-surface**, **n2-pos**, **n2-lemma**, **n2-det** and **parser-vote** on examples that follow the verb. *FN-best-no-sem* uses **v-surface**, **v-pos**, **v-lemma**, **subcategorization**, **path** (reduced POS), **position**, **n1-preposition**, **n1-surface**, **n1-pos**, **n1-lemma-mpf**, **n1-parent**, **p**, **n2-surface**, **n2-pos**, **n2-lemma-mpf**, **n2-det**, **parser-vote**, **count-ratio** and **pp-count** on all examples. *FN-best-sem* uses the same set of features as *FN-best-no-sem* plus **v-frame** and **n1-sr**.

TB2-basic uses **v-surface**, **n1-surface-mpf**, **p** and **n2-surface-mpf**. *TB2-best-no-www* uses **v-surface**, **v-pos**, **v-lemma**, **subcategorization**, **path** (reduced POS), **n1-preposition**, **n1-surface**, **n1-mpf**, **n1-pos**, **n1-lemma**, **n1-np-label**, **p**, **n2-surface**, **n2-mpf** and **n1-p-distance**. *TB2-best* also uses **count-ratio** and **pp-count**.

RRR-basic uses **v-surface**, **n1-surface**, **p** and **n2-surface**. *RRR-basic-mpf* uses **v-surface**, **n1-surface-mpf**, **p** and **n2-surface-mpf**.

On the FN dataset, all features except **v-voice** have a positive contribution to the system (**n2-det**, choice between semantic vs. thematic role and how should morphological processing be applied is questionable). The negative impact for the **v-voice** feature may be explained by the fact that the only situation in which it may potentially help is extremely rare: passive voice and the agent headed by *"by"* appears after another argument of the verb (i.e.: *"The painting was presented to the audience by its author."*). Moreover the PP-attachment based on the preposition *"by"* is not highly ambiguous; as seen in Table 3 in the FrameNet dataset, 88% of the *"by"* ambiguity instances are verb-attachments.

The experiment with the highest cross-validation accuracy has an accuracy of 92.85% on the test data. The equivalent experiment that doesn't include manually annotated semantic information has an accuracy of 91.79% on the test data.

On TB2 dataset, the results are close to the results obtained on the FrameNet corpus, although the distribution of noun and verb attachment differs considerably between the two data sets (70.28% are verb-attachments in FN2 and 35.71% in TB2). The best accuracy in cross-validation is 92.92%, which leads to an accuracy on test set of 93.62%.

7 Comparison with previous work

Because we couldn't use the standard dataset used in PP-attachment resolution (Ratnaparkhi's), we implemented back-off algorithm developed by Collins and Brooks (1995) and applied it to our TB2 dataset. Both RRR and TB2 datasets are extracted from Penn Treebank. This algorithm, trained on TB2 training set, obtains an accuracy on TB2 test set of 86.1% (85.8% when no morphological processing is applied). The same algorithm provides an accuracy on RRR dataset of 84.5% (84.1% without morphological processing). The difference in accuracy between the two datasets is 1.6% (1.7% without morphological processing when using Collins and Brooks's algorithm.

The difference in accuracy between a SVM model applied to RRR dataset (*RRR-basic* experiment) and the same experiment applied to TB2 dataset (*TB2-*

Description	Accuracy	Data	Extra Supervision
Always noun	55.0	RRR	
Most likely for each P	72.19	RRR	
Most likely for each P	**72.30**	**TB2**	
Most likely for each P	**81.73**	**FN**	
Average human, headwords (Ratnaparkhi et al., 1994)	88.2	RRR	
Average human, whole sentence (Ratnaparkhi et al., 1994)	93.2	RRR	
Maximum Likelihood-based (Hindle and Rooth, 1993)	79.7	AP	
Maximum entropy, words (Ratnaparkhi et al., 1994)	77.7	RRR	
Maximum entropy, words & classes (Ratnaparkhi et al., 1994)	81.6	RRR	
Decision trees (Ratnaparkhi et al., 1994)	77.7	RRR	
Transformation-Based Learning (Brill and Resnik, 1994)	81.8		WordNet
Maximum-Likelihood based (Collins and Brooks, 1995)	84.5	RRR	
Maximum-Likelihood based (Collins and Brooks, 1995)	**86.1**	**TB2**	
Decision trees & WSD (Stetina and Nagao, 1997)	88.1	RRR	WordNet
Memory-based Learning (Zavrel et al., 1997)	84.4	RRR	LexSpace
Maximum entropy, unsupervised (Ratnaparkhi, 1998)	81.9		
Maximum entropy, supervised (Ratnaparkhi, 1998)	83.7	RRR	
Neural Nets (Alegre et al., 1999)	86.0	RRR	WordNet
Boosting (Abney et al., 1999)	84.4	RRR	
Semi-probabilistic (Pantel and Lin, 2000)	84.31	RRR	
Maximum entropy, ensemble (McLauchlan, 2001)	85.5	RRR	LSA
SVM (Vanschoenwinkel and Manderick, 2003)	84.8	RRR	
Nearest-neighbor (Zhao and Lin, 2004)	86.5	RRR	DWS
FN dataset, w/o semantic features (*FN-best-no-sem*)	**91.79**	**FN**	**PR-WWW**
FN dataset, w/ semantic features (*FN-best-sem*)	**92.85**	**FN**	**PR-WWW**
TB2 dataset, best feature set (*TB2-best*)	**93.62**	**TB2**	**PR-WWW**

Table 5: Accuracy of PP-attachment ambiguity resolution (our results in bold)

basic experiment) is 2.9%. Also, the baseline – the most probable PP type for each preposition – is approximately the same for the two datasets (72.19% on RRR and 72.30% on TB2).

One may hypothesize that the majority of the algorithms for PP-attachment disambiguation obtain no more than 4% increase in accuracy on the TB2 compared to the results on the RRR dataset. One important difference between the two datasets is the size – 20,801 training examples in RRR vs. 54,629 training examples in TB2. We plan to implement more algorithms described in literature in order to verify this statement.

Table 5 summarizes the results in PP-attachment ambiguity resolution found in literature along with our best results.

Other acronyms used in this table:

- AP – dataset of 13 million word sample of Associated Press news stories from 1999 (Hindle and Rooth, 1993).
- LexSpace - Lexical Space – a method to measure the similarity of the words (Zavrel et al., 1997).
- LSA – Latent Semantic Analysis – measure the lexical preferences between a preposition and a noun or a verb (McLauchlan, 2001)
- DWS – Distributional Word Similarity. Words that tend to appear in the same contexts tend to have similar meanings (Zhao and Lin, 2004)
- PR-WWW – the probability ratio between verb-preposition-noun and noun-preposition-noun constructs measured using World Wide Web searching.

8 Conclusions

The Penn Treebank-II results indicate that the new features used for the disambiguation of PP-attachment provide a very substantial improvement in accuracy over the base line (from 87.48% to 93.62%). This represents an absolute improvement of approximately 6.14%, equivalent to a 49% error drop. The performance of the system on Penn Treebank-II exceeds the reported human expert performance on Penn Treebank-I (Ratnaparkhi et al., 1994) by about 0.4%. A significant improvement comes from the unsupervised information collected from a very large corpus; this method proved to be efficient to overcome the data sparseness problem.

By analyzing the results from the FrameNet dataset, we conclude that the contribution of the gold semantic features (frame and semantic role) is significant (1.05% difference in accuracy; 12.8% reduction in the error). We will further investigate this issue by replacing gold semantic information with automatically detected semantic information. Our additional lexico-syntactic features increase the accuracy of the system from 86.44% to 89.61% for PPs following the verb. This suggests that on the FrameNet dataset the proposed syntactic features have a considerable impact on the accuracy.

The best TB2 feature set is approximately the same as the best FN feature set in spite of the differences between the datasets (Parse trees: TB2 – gold standard; FN – automatically generated. PP-attachment ambiguity identification: TB2 – parse trees; FN – a combination of trees and FE annotation. Data source: TB2 – WSJ articles; FN – BNC). This fact suggests that the selected feature sets do not exploit particularities of the datasets and that the features are relevant to the PP-attachment ambiguity problem.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and PP Attachment. In *Proceedings of EMNLP/VLC-99*, pages 38–45.

Martha A. Alegre, Josep M. Sopena, and Agusti Lloberas. 1999. Pp-attachment: A committee machine approach. In *Proceedings of EMNLP/VLC-99*, pages 231–238.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational Linguistics*, pages 86–90.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th conference on Computational Linguistics*, pages 1198–1204.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL-2000*, pages 132–139.

Michael Collins and James Brooks. 1995. Prepositional Phrase Attachment through a Backed-Off Model. In *Proceedings of the Thirds Workshop on Very Large Corpora*, pages 27–38.

Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Donald Hindle and Mats Rooth. 1993. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19(1):103–120.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mark McLauchlan. 2001. Maximum Entropy Models and Prepositional Phrase Ambiguity. Master's thesis, University of Edinburgh.

Marian G. Olteanu. 2004. Prepositional Phrase Attachment ambiguity resolution through a rich syntactic, lexical and semantic set of features applied in support vector machines learner. Master's thesis, University of Texas at Dallas.

Patrick Pantel and Dekang Lin. 2000. An unsupervised approach to Prepositional Phrase Attachment using contextually similar words. In *Proceedings of the 38th Meeting of the Association for Computational Linguistic*, pages 101–108.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2003. Semantic Role Parsing: Adding Semantic Structure to Unstructured Text. In *Proceedings of the International Conference on Data Mining*, pages 629–632.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the Human Language Technology Workshop*, pages 250–255.

Adwait Ratnaparkhi. 1998. Statistical Models for Unsupervised Prepositional Phrase Attachment. In *Proceedings of the 36th conference on Association for Computational Linguistics*, pages 1079–1085.

Jiri Stetina and Makoto Nagao. 1997. Corpus based PP attachment ambiguity resolution with a semantic dictionary. In *Proceedings of the Fifth Workshop on Very Large Corpora*, pages 66–80.

Bram Vanschoenwinkel and Bernard Manderick. 2003. A weighted polynomial information gain kernel for resolving Prepositional Phrase attachment ambiguities with Support Vector Machines. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 133–140.

Martin Volk. 2001. Exploiting the WWW as a corpus to resolve PP attachment ambiguities. In *Proceedings of Corpus Linguistics*, pages 601–606.

Jakub Zavrel, Walter Daelemans, and Jorn Veenstra. 1997. Resolving PP attachment Ambiguities with Memory-Based Learning. In *Proceedings of CoNLL-97*, pages 136–144.

Shaojun Zhao and Dekang Lin. 2004. A Nearest-Neighbor Method for Resolving PP-Attachment Ambiguity. In *Proceedings of IJCNLP-04*.

Compiling Comp Ling: Practical Weighted Dynamic Programming and the Dyna Language[*]

Jason Eisner and **Eric Goldlust** and **Noah A. Smith**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{jason,goldlust,nasmith}@cs.jhu.edu

Abstract

Weighted deduction with aggregation is a powerful theoretical formalism that encompasses many NLP algorithms. This paper proposes a declarative specification language, Dyna; gives general *agenda-based* algorithms for computing weights and gradients; briefly discusses Dyna-to-Dyna program transformations; and shows that a first implementation of a Dyna-to-C++ compiler produces code that is efficient enough for real NLP research, though still several times slower than hand-crafted code.

1 Introduction

In this paper, we generalize some modern probabilistic parsing techniques to a broader class of weighted deductive algorithms. Our implemented system encapsulates these implementation techniques behind a clean interface—a small high-level specification language, Dyna, which compiles into C++ classes. This system should help the HLT community to experiment more easily with new models and algorithms.

1.1 Dynamic programming as deduction

The "parsing as deduction" framework (Pereira and Warren, 1983) is now over 20 years old. It provides an elegant notation for specifying a variety of parsing algorithms (Shieber et al., 1995), including algorithms for probabilistic or other semiring-weighted parsing (Goodman, 1999). In the parsing community, new algorithms are often stated simply as a set of deductive inference rules (Sikkel, 1997; Eisner and Satta, 1999).

It is also straightforward to specify other NLP algorithms this way. Syntactic MT models, language models, and stack decoders can be easily described using deductive rules. So can operations on finite-state and infinite-state machines.

[*]We thank Joshua Goodman, David McAllester, and Paul Ruhlen for useful early discussions; pioneer users Markus Dreyer, David Smith, and Roy Tromble for their feedback and input; John Blatz for discussion of program transformations; and several reviewers for useful criticism. This work was supported by NSF ITR grant IIS-0313193, ONR MURI grant N00014-01-1-0685, and a Hertz Foundation fellowship to the third author. The views expressed are not necessarily endorsed by the sponsors.

1.2 The role of toolkits

One might regard deductive inference as merely a helpful perspective for teaching old algorithms and thinking about new ones, linking NLP to logic and classical AI. Real implementations would then be carefully hand-coded in a traditional language.

That was the view ten years ago of finite-state machines—that FSMs were part of the theoretical backbone of CL, linking the field to the theory of computation. Starting in the mid-1990's, however, finite-state methods came to the center of *applied* NLP as researchers at Xerox, AT&T, Groningen and elsewhere improved the expressive power of FSMs by moving from automata to transducers, adding semiring weights, and developing powerful new regular-expression operators and algorithms for these cases. They also developed software. Karttunen et al. (1996) built an FSM toolkit that allowed construction of morphological analyzers for many languages. Mohri et al. (1998) built a *weighted* toolkit that implemented novel algorithms (e.g., weighted minimization, on-the-fly composition) and scaled up to handle large-vocabulary continuous ASR. At the same time, renewed community-wide interest in shallow methods for information extraction, chunking, MT, and dialogue processing meant that such off-the-shelf FS toolkits became the core of diverse systems used in cutting-edge research.

The weakness of FSMs, of course, is that they are only finite-state. One would like something like AT&T's FSM toolkit that also handles the various formalisms now under consideration for lexicalized grammars, non-context-free grammars, and syntax-based MT—and hold the promise of extending to other formalisms and applications not yet imagined.

We believe that deductive inference should play the role of regular expressions and FSMs, providing the theoretical foundation for such an effort. Many engineering ideas in the field can be regarded, we

```
1. :- double item=0.                                      % declares that all item values are doubles, default is 0
2. constit(X,I,K) += rewrite(X,W) * word(W,I,K).          % a constituent is either a word ...
3. constit(X,I,K) += rewrite(X,Y,Z) * constit(Y,I,J) * constit(Z,J,K).  % ... or a combination of two adjacent subconstituents
4. goal += constit("s",0,N) whenever ?ends_at(N).         % a parse is any s constituent that covers the input string
```

Figure 1: A probabilistic CKY parser written in Dyna. Axioms are in boldface.

believe, as ideas for how to specify, transform, or compile systems of inference rules.

2 A Language for Deductive Systems

Any toolkit needs an interface. For example, FS toolkits offer a regular expression language. We propose a simple but Turing-complete language, Dyna, for specifying weighted deductive-inference algorithms. We illustrate it here by example; see `http://dyna.org` for more details and a tutorial.

The short Dyna program in Fig. 1 expresses the inside algorithm for PCFGs (i.e., the probabilistic generalization of CKY recognition). Its 3 **inference rules** schematically specify many *equations*, over an arbitrary number of unknowns. This is possible bcause the unknowns (**items**) have *structured names* (**terms**) such as constit("s",0,3). They resemble typed variables in a C program, but we use **variable** instead to refer to the capitalized identifiers X, I, K, ...in lines 2–4. Each rule gives a **consequent** on the left-hand side of the +=, which can be built by combining the **antecedents** on the right-hand side.[1]

Lines 2–4 are equational schemas that specify how to compute the value of items such as constit("s",0,3) from the values of other items. Using the summation operator +=, lines 2–3 say that for any X, I, and K, constit(X,I,K) is defined by summing over the remaining variables, as $\sum_{\mathsf{W}}$ rewrite(X,W)*word(W,I,K) $+$ $\sum_{\mathsf{Y,Z,J}}$ rewrite(X,Y,Z)*constit(Y,I,J)*constit(Z,J,K). For example, constit("s",0,3) is a sum of quantities such as rewrite("s", "np", "vp")*constit("np",0,1)*constit("vp",1,3). The whenever operator in line 4 specifies a **side condition** that restricts the set of expressions in the sum (i.e., only when N is the sentence length).

To fully define the system of equations, non-default values (in this case, non-zero values) should be **asserted** for some **axioms** at runtime. (Axioms, shown in bold in Fig. 1, are items that never appear as a consequent.) If the PCFG contains a rewrite rule np $\rightarrow$ Mary with probability $p(\mathsf{Mary} \mid \mathsf{np})$=0.005, the user should assert that rewrite("np", "Mary") has value 0.005. If the input is *John loves Mary*, values of 1 should be asserted for word("John",0,1), word("loves",1,2), word("Mary",2,3), and ends_at(3).

Given the axioms as base cases, the equations in Fig. 1 enable deduction of values for other items. The value of the **theorem** constit("s",0,3) will be the inside probability $\beta_{\mathsf{s}}(0,3)$,[2] and the value of goal will be the total probability of all parses.

If one replaces += by max= throughout, then constit("s",0,3) will accumulate the maximum rather than the sum of these quantities, and goal will accumulate the probability of the *best* parse.

With different input, the same program carries out lattice parsing. Simply assert axioms that correspond to (weighted) lattice arcs, such as word("John",17,50), where 17 and 50 are arbitrary terms denoting states in the lattice. It is also quite straightforward to lexicalize the nonterminals or extend to synchronous grammars.

A related context-free parsing strategy, shown in Fig. 2, is Earley's algorithm. These equations illustrate nested terms such as lists. The side condition in line 2 prevents building any constituent until one has built a left context that calls for it.

3 Relation to Previous Work

There is a large relevant literature. Some of the well-known CL papers, notably Goodman (1999), were already mentioned in section 1.1. Our project has three main points of difference from these.

First, we provide an efficient, scalable, open-source implementation, in the form of a compiler from Dyna to C++ classes. (Related work is in §7.2.) The C++ classes are efficient and easy to use, with statements such as c[rewrite("np",2,3)]=0.005 to assert axiom values into a chart named c (i.e., a deduc-

[1]Much of our notation and terminology comes from logic programming: term, variable, inference rule, antecedent/consequent, assert/retract, axiom/theorem.

[2]That is, the probability that s would stochastically rewrite to the first three words of the input. If this can happen in more than one way, the probability sums over multiple derivations.

```
1. need("s",0) = 1.                                                        % begin by looking for an s that starts at position 0
2. constit(Nonterm/Needed,I,I) += rewrite(Nonterm,Needed) whenever ?need(Nonterm, I).     % traditional predict step
3. constit(Nonterm/Needed,I,K) += constit(Nonterm/cons(W,Needed),I,J) * word(W,J,K).      % traditional scan step
4. constit(Nonterm/Needed,I,K) += constit(Nonterm,cons(X,Needed),I,J) * constit(X/nil,J,K).  % traditional complete step
5. goal += constit("s"/nil,0,N) whenever ?ends_at(N).                 % we want a complete s constituent covering the sentence
6. need(Nonterm,J) += constit(_/cons(Nonterm,_),_,J).       % Note: underscore matches anything (anonymous wildcard)
```

Figure 2: An Earley parser that recovers inside probabilities (Earley, 1970; Stolcke, 1995). The rule np $\rightarrow$ det n should be encoded as the axiom rewrite("np",cons("det",cons("n",nil))), a nested term. "np"/Needed is the label of a partial np constituent that is still missing the *list* of subconstituents in Needed. need("np",3) is derived if some partial constituent seeks an np subconstituent starting at position 3. As in Fig. 1, lattice parsing comes for free, as does training.

tive database) and expressions like c[goal] to extract the values of the resulting theorems, which are computed as needed. The C++ classes also give access to the proof forest (e.g., the forest of parse trees), and integrate with parameter optimization code.

Second, we fully generalize the agenda-based strategy of Shieber et al. (1995) to the weighted case—in particular supporting a *prioritized* agenda. That allows probabilities to guide the search for the best parse(s), a crucial technique in state-of-the-art context-free parsers.[3] We also give a "reverse" agenda algorithm to compute gradients or outside probabilities for parameter estimation.

Third, regarding weights, the Dyna language is designed to express systems of arbitrary, heterogeneous equations over item values. In previous work such as (Goodman, 1999; Nederhof, 2003), one only specifies the inference rules as unweighted Horn clauses, and then weights are added automatically in a standard way: all values have the same type $\mathcal{W}$, and all rules transform to equations of the form $c \oplus= a_1 \otimes a_2 \otimes \cdots \otimes a_k$, where $\oplus$ and $\otimes$ give $\mathcal{W}$ the structure of a semiring.[4] In Dyna one writes these equations explicitly in place of Horn clauses (Fig. 1). Accordingly, *heterogeneous* Dyna programs, to be supported soon by our compiler, will allow items of different types to have values of different types, computed by different aggregation operations over arbitrary right-hand-side expressions. This allows specification of a wider class of algorithms from NLP and elsewhere (e.g., minimum expected loss decoding, smoothing formulas, neural networks, game tree analysis, and constraint programming). Although §4 and §5 have space to present only techniques for the semiring case, these can be generalized.

Our approach may be most closely related to deductive databases, which even in their heyday were apparently ignored by the CL community (except for Minnen, 1996). Deductive database systems permit inference rules that can derive new database facts from old ones.[5] They are essentially declarative logic programming languages (with restrictions or extensions) that are—or could be—implemented using efficient database techniques. Some implemented deductive databases such as CORAL (Ramakrishnan et al., 1994) and LOLA (Zukowski and Freitag, 1997) support aggregation (as in Dyna's +=, log+=, max=, ...), although only "stratified" forms of it that exclude unary CFG rule cycles.[6] Ross and Sagiv (1992) (and in a more restricted way, Kifer and Subrahmanian, 1992) come closest to our notion of attaching aggregable values to terms.

Among deductive or other database systems, Dyna is perhaps unusual in that its goal is not to support transactional databases or *ad hoc* queries, but rather to serve as an abstract layer for specifying an algorithm, such as a dynamic programming (DP) algorithm. Thus, the Dyna program already implicitly or explicitly specifies all queries that will be needed. This allows compilation into a hard-coded C++ implementation. The compiler's job is to support these queries by laying out and indexing the database re-

[3]Previous treatments of weighted deduction have used an agenda only for an unweighted parsing phase (Goodman, 1999) or for finding the single best parse (Nederhof, 2003). Our algorithm works in arbitrary semirings, including non-idempotent ones, taking care to avoid double-counting of weights and to handle side conditions.

[4]E.g., the inside algorithm in Fig. 1 falls into Goodman's framework, with $\langle \mathcal{W}, \oplus, \otimes \rangle = \langle \mathbb{R}_{\geq 0}, +, * \rangle$—the PLUSTIMES semiring. Because $\otimes$ distributes over $\oplus$ in a semiring, computing goal is equivalent to an aggregation over many separate parse trees. That is not the case for heterogeneous programs.

[5]Often they use some variant of the *unweighted* agenda-based algorithm, which is known in that community as "semi-naive bottom-up evaluation."

[6]An unweighted parser was implemented in an earlier version of LOLA (Specht and Freitag, 1995).

lations in memory[7] in a way that resembles hand-designed data structures for the algorithm in question. The compiler has many choices to make here; we ultimately hope to implement feedback-directed optimization, using profiled sample runs on typical data. For example, a sparse grammar should lead to different strategies than a dense one.

4 Computing Theorem Values

Fig. 1 specifies a set of equations but not how to solve them. Any declarative specification language must be backed up by a solver for the class of specifiable problems. In our continuing work to develop a range of compiler strategies for arbitrary Dyna programs, we have been inspired by the CL community's experience in building efficient parsers.

In this paper and in our current implementation, we give only the algorithms for what we call *weighted dynamic programs*, in which all axioms and theorems are variable-free. This means that a consequent may only contain variables that already appear elsewhere in the rule. We further restrict to semiring-weighted programs as in (Goodman, 1999). But with a few more tricks not given here, the algorithms can be generalized to a wider class of heterogeneous weighted logic programs.[8]

4.1 Desired properties

Computation is triggered when the user requests the value of one or more particular items, such as goal. Our algorithm must have several properties in order to substitute for manually written code.

Soundness. The algorithm cannot be guaranteed to terminate (since it is possible to write arbitrary Turing machines in Dyna). However, if it does terminate, it should return values from a valid model of the program, i.e., values that simultaneously satisfy all the equations expressed by the program.

Reasonable completeness. The computation should indeed terminate for programs of interest to the NLP community, such as parsing under a probabilistic grammar—even if the grammar has left recursion, unary rule cycles, or ϵ-productions. This appears to rule out pure top-down ("backward-chaining") approaches.

Efficiency. Returning the value of goal should do only as much computation as necessary. To return goal, one may not need to compute the values of *all* items.[9] In particular, finding the best parse should not require finding all parses (in contrast to Goodman (1999) and Zhou and Sato (2003)). Approximation techniques such as pruning and best-first search must also be supported for practicality.

```
1.  for each axiom $a$, set $agenda[a] :=$ value of axiom $a$
2.  while there is an item $a$ with $agenda[a] \neq \mathbf{0}$
3.      (* remove an item from the agenda and move its value to the chart *)
4.      choose such an $a$
5.      $\Delta := agenda[a];\ agenda[a] := \mathbf{0}$
6.      $old := chart[a];\ chart[a] := chart[a] \oplus \Delta$
7.      if $chart[a] \neq old$            (* only propagate actual changes *)
8.          (* compute new resulting updates and place them on the agenda *)
9.          for each inference rule "$c \mathrel{\oplus}= a_1 \otimes a_2 \otimes \cdots \otimes a_k$"
10.             for $i$ from 1 to $k$
11.                 for each way of instantiating the rule's variables
                    such that $a_i = a$
12.                     $agenda[c] \mathrel{\oplus}= \bigotimes_{j=1}^{k} \begin{cases} old & \text{if } j < i \text{ and } a_j = a \\ \Delta & \text{if } j = i \\ chart[a_j] & \text{otherwise} \end{cases}$
                        (* can skip this line if any multiplicand is $\mathbf{0}$ *)
```

Figure 3: Weighted agenda-based deduction in a semiring, without side conditions (see text).

4.2 The agenda algorithm

Our basic algorithm (Fig. 3) is a weighted agenda-based algorithm that works only with rules of the form $c \mathrel{\oplus}= a_1 \otimes a_2 \otimes \cdots \otimes a_k$. $\otimes$ must distribute over $\oplus$. Further, the default value for items (line 1 of Fig. 1) must be the semiring's zero element, denoted $\mathbf{0}$.[10]

Agenda-based deduction maintains two indexed data structures: the **agenda** and the **chart**. $chart[a]$ stores the current value of item a. The agenda holds future work that arises from assertions or from previous changes to the chart: $agenda[a]$ stores an incremental update to be added (using $\oplus$) to $chart[a]$ in future. If $chart[a]$ or $agenda[a]$ is not stored, it is

[7]Some relations might be left unmaterialized and computed on demand, with optional memoization and flushing of memos.

[8]Heterogeneous programs may propagate non-additive updates, which arbitrarily modify one of the inputs to an aggregation. Non-dynamic programs require non-ground items in the chart, complicating both storage and queries against the chart.

[9]This also affects completeness, as it sometimes enables the computation of goal to terminate even if the program as a whole contains some irrelevant non-terminating computation. Even in practical cases, the runtime of computing all items is often prohibitive, e.g., proportional to n^6 or worse for a dense tree-adjoining grammar or synchronous grammar.

[10]It satisfies $x \oplus \mathbf{0} = x, x \otimes \mathbf{0} = \mathbf{0}$ for all x. Also, this algorithm requires $\otimes$ to distribute over $\oplus$. Dyna's semantics requires $\oplus$ to be associative and commutative.

taken to be the default $\mathbf{0}$.

When item a is removed from the agenda, its chart weight is updated by the increment value. This change is then propagated to other items c, via rules of the form $c \oplus= \cdots$ with a on the right-hand-side. The resulting changes to c are placed back on the agenda and carried out only later.

The unweighted agenda-based algorithm (Shieber et al., 1995) may be regarded as the case where $\langle \mathcal{W}, \oplus, \otimes \rangle = \langle \{T, F\}, \vee, \wedge \rangle$. It has previously been generalized (Nederhof, 2003) to the case $\langle \mathcal{W}, \oplus, \otimes \rangle = \langle \mathbb{R}_{\geq 0}, \max, + \rangle$. In Fig. 3, we make the natural further generalization to any semiring.

How is this a further generalization? Since $\oplus$ (unlike $\vee$ and max) might not be idempotent, we must take care to avoid erroneous double-counting if the antecedent a combines with, or produces, another copy of itself.[11] For instance, if the input contains ϵ words, line 2 of Fig. 1 may get instantiated as constit("np",5,5) += rewrite("np","np","np") * constit("np",5,5) * constit("np",5,5). This is why we save the old values of *agenda*$[a]$ and *chart*$[a]$ as Δ and *old*, and why line 12 is complex.

4.3 Side conditions

We now extend Fig. 3 to handle Dyna's side conditions, i.e., rules of the form $c \oplus=$ *expression* whenever *boolean-expression*. We discuss only the simple side conditions treated in previous literature, which we write as $c \oplus= a_1 \otimes a_2 \otimes \cdots \otimes a_{k'}$ whenever $?b_{k'+1}$ & $\cdots$ & $?b_k$. Here, $?b_j$ is true or false according to whether there exists an *unweighted* proof of b_j.

Again, what is new here? Nederhof (2003) considers only max= with a uniform-cost agenda discipline (see §4.5), which guarantees that no item will be removed more than once from the agenda. We wish to support other cases, so we must take care that a second update to a_i will not retrigger rules of which a_i is a side condition.

For simplicity, let us reformulate the above rule as $c \oplus= a_1 \otimes a_2 \otimes \cdots \otimes a_{k'} \otimes ?b_{k'+1} \otimes \cdots \otimes ?b_k$, where $?b_i$ is now treated as having value $\mathbf{0}$ or $\mathbf{1}$ (the identity for $\otimes$) rather than false or true respectively.

[11] An agenda update that increases x by 0.3 will increase $r * x * x$ by $r * (0.6x + 0.09)$. Hence, the rule x += $r * x * x$ must propagate a new increase of that size to x, via the agenda.

We may now use Fig. 3, but now any a_j might have the form $?b_j$. Then in line 12, *chart*$[a_j]$ will be *chart*$[?b_j]$, which is defined as $\mathbf{1}$ or $\mathbf{0}$ according to whether *chart*$[b_j]$ is stored (i.e., whether b_j has been derived). Also, if $a_i = ?a$ at line 11 (rather than $a_i = a$), then Δ in line 12 is replaced by $\Delta ?$, where we have set $\Delta ? :=$ *chart*$[?a]$ at line 5.

4.4 Convergence

Whether the agenda algorithm halts depends on the Dyna program and the input. Like any other Turing-complete language, Dyna gives you enough freedom to write undesirable programs.

Most NLP algorithms do terminate, of course, and this remains true under the agenda algorithm. For typical algorithms, only finitely many different items (theorems) can be derived from a given finite input (set of axioms).[12] This ensures termination if one is doing unweighted deduction with $\langle \mathcal{W}, \oplus, \otimes \rangle = \langle \{T, F\}, \vee, \wedge \rangle$, since the test at line 7 ensures that no item is processed more than once.[13]

The same test ensures termination if one is searching for the best proof or parse with (say) $\langle \mathcal{W}, \oplus, \otimes \rangle = \langle \mathbb{R}_{\geq 0}, \min, + \rangle$, where values are negated log probabilities. Positive-weight cycles will not affect the min. (Negative-weight cycles, however, would correctly cause the computation to diverge; these do not arise with probabilities.)

If one is using $\langle \mathcal{W}, \oplus, \otimes \rangle = \langle \mathbb{R}_{\geq 0}, +, * \rangle$ to compute the total weight of all proofs or parses, as in the inside algorithm, then Dyna must solve a system of nonlinear equations. The agenda algorithm does this by iterative approximation (propagating updates around any cycles in the proof graph until numerical convergence), essentially as suggested by Stolcke (1995) for the case of Earley's algorithm.[14] Again, the computation may diverge.

[12] This holds for all Datalog programs, for instance.

[13] This argument does not hold if Dyna is used to express programs outside the semiring. In particular, one can write instances of SAT and other NP-hard constraint satisfaction problems by using cyclic rules *with negation* over finitely many boolean-valued items (Niemelä, 1998). Here the agenda algorithm can end up flipping values forever between false and true; a more general solver would have to be called in order to find a stable model of a SAT problem's equations.

[14] Still assuming the number of items is finite, one could in principle materialize the system of equations and call a dedicated numerical solver. In some special cases only a linear solver is needed: e.g., for unary rule cycles (Stolcke, 1995), or ϵ-cycles in FSMs (Eisner, 2002).

One can declare the conditions under which items of a particular type (constit or goal) should be treated as having converged. Then asking for the value of goal will run the agenda algorithm not until the agenda is empty, but only until *chart*[goal] has converged by this criterion.

4.5 Prioritization

The order in which items are chosen at line 4 does not affect the soundness of the agenda algorithm, but can greatly affect its speed. We implement the agenda as a priority queue whose priority function may be specified by the user.[15]

Charniak et al. (1998) and Caraballo and Charniak (1998) showed that, when seeking the best parse (using min= or max=), *best-first* parsing can be extremely effective. Klein and Manning (2003a) went on to describe admissible heuristics and an A* framework for parsing. For A* in our general framework, the priority of item a should be an estimate of the value of the best proof of goal that uses a. (This non-standard formulation is carefully chosen.[16]) If so, goal is guaranteed to converge the very first time it is selected from the priority-queue agenda.

Prioritizing "good" items first can also be useful in other circumstances. The inside-outside training algorithm requires one to find all parses, but finding the high-probability parses first allows one to ignore the rest by "early stopping."

In all these schemes (even A*), processing promising items as soon as possible risks having to reprocess them if their values change later. Thus, this strategy should be balanced against the "topological sort" strategy of waiting to process an item until its value has (probably) converged.[17] Ultimately we hope to *learn* priority functions that effectively balance these two strategies (especially in the context of early stopping).

[15]At present by writing a C++ function; ultimately within Dyna, by defining items such as priority(constit("s",0,3)).

[16]It is correct for proofs that incorporate two copies of a's value, or—more important—no copies of a's value because a is a side condition. Thus, it recognizes that a low-probability item must have *high* priority if it could be used as a side condition in a higher-probability parse (though this cannot happen for the side conditions derived by the magic templates transformation (§6)). Note also that a's own value (Nederhof, 2003) might not be an optimistic estimate, if negative weights are present.

[17]In parsing, for example, one often processes narrower constituents before wider ones. But such strategies do not always exist, or break down in the presence of unary rule cycles, or cannot be automatically found. Goodman's (1999) strategy of building all items and sorting them before computing any weights is wise only if one genuinely wants to build all items.

4.6 Matching, indexing, and interning

The crucial work in Fig. 3 occurs in the iteration over instantiated rules at lines 9–11. In practice, we restructure this triply nested loop as follows, where each line retains the variable bindings that result from the unification in the previous line:

```
9.  **for** each antecedent pattern $a_i$ that appears in some
    program rule $r$ and unifies with $a$
10.     **for** each way of simultaneously unifying $r$'s remaining antecedent patterns $a_1, \ldots a_{i-1}, a_{i+1}, \ldots a_k$
        with items that may have non-**0** value in the chart
11.         construct $r$'s consequent $c$    (* all vars are bound *)
```

Our implementation of line 9 tests a against all of the antecedent patterns at once, using a tree of simple "if" tests (generated by the Dyna-to-C++ compiler) to share work across patterns. As an example, $a =$ constit("np",3,8) will match two antecedents at line 3 of Fig. 1, but will fail to match in line 4. Because a is variable-free (for DPs), a full unification algorithm is not necessary, even though an antecedent pattern can contain repeated variables and nested subterms.

Line 10 rapidly looks up the rule's other antecedents using *indices* that are automatically maintained on the chart. For example, once constit("np",4,8) has matched antecedent 2 of line 3 of Fig. 1, the compiled code consults a maintained list of the chart constituents that start at position 8 (i.e., items of the form constit(Z,8,K) that have already been derived). Suppose one of these is constit("vp",8,15): then the code finds the rule's remaining antecedent by consulting a list of items of the form rewrite(X,"np","vp"). That leads it to construct consequents such as constit("s",4,15) at line 11.

By default, equal terms are represented by equal pointers. While this means terms must be "interned" when constructed (requiring hash lookup), it enforces structure-sharing and allows any term to be rapidly copied, hashed, or equality-tested without dereferencing the pointer.[18]

Each of the above paragraphs conceals many decisions that affect runtime. This presents future opportunities for feedback-directed optimization, where profiled runs on typical data influence the compiler.

[18]The compiled code provides garbage collection on the terms; this is important when running over large datasets.

5 Computing Gradients

The value of goal is a *function* of the axioms' values. If the function is differentiable, we may want to get its gradient with respect to its parameters (the axiom values), to aid in numerically optimizing it.

5.1 Gradients by symbolic differentiation

The gradient computation can be derived from the original by a program transformation. For each item a in the original program—in particular, for each axiom—the new program will also compute a new item $\mathsf{g}(a)$, whose value is $\partial\mathsf{goal}/\partial a$.

Thus, given weighted axioms, the new program computes both goal and $\nabla\mathsf{goal}$. An optimization algorithm such as conjugate gradient can use this information to tune the axiom weights to maximize goal. An alternative is the EM algorithm (Dempster et al., 1977) for probabilistic generative models such as PCFGs. Luckily the same program serves, since for such models, the E count (expected count) of an item a can be found as $a \cdot \mathsf{g}(a)/\mathsf{goal}$. In other words, the inside-outside algorithm has the same structure as computing the function and its gradient.

The GRADIENT transformation is simple. For example,[19] given a rule c += $a_1 * a_2 * \cdots * a_{k'}$ whenever $?b_{k'+1}$ & $\cdots$ & $?b_k$, we add a new rule $\mathsf{g}(a_i)$ += $\mathsf{g}(c) * a_1 * \cdots * a_{i-1} * a_{i+1} * \cdots * a_{k'}$ whenever $?a_i$, for each $i = 1, 2, ..., k'$. (The original rule remains, since we need inside values to compute outside values.) This strategy for computing the gradient $\partial\mathsf{goal}/\partial a$ via the chain rule is an example of automatic differentiation in the reverse mode (Griewank and Corliss, 1991), known in the neural network community as back-propagation.

5.2 Gradients by back-propagation

However, what if goal might be computed only approximately, by early stopping before convergence (§4.5)? To avoid confusing the optimizer, we want the *exact* gradient of the *approximate* function.

To do this, we "unwind" the computation of goal, *undoing* the value updates while building up the gradient values. The idea is to differentiate an "unrolled" version of the original computation (Williams and Zipser, 1989), in which an item at

[19]More generally, $\mathsf{g}(a_i) = \partial\mathsf{goal}/\partial a_i = \sum_c \partial\mathsf{goal}/\partial c \cdot \partial c/\partial a_i = \sum_c \mathsf{g}(c) \cdot \partial c/\partial a_i$ by the chain rule.

```
1.  for each $a$, $gchart[a] := 0$ and $gagenda[a] := 0$
        (* respectively hold $\partial\mathsf{goal}/\partial chart[a]$ and $\partial\mathsf{goal}/\partial agenda[a]$ *)
2.  $gchart[\mathsf{goal}] := 1$
3.  for each $\langle a, \Delta, old\rangle$ triple that was considered at line 8
    of Fig. 3, but in the reverse order                    (* $\Delta$ is agenda[a] *)
4.    $\Gamma := gchart[a]$                      (* will accumulate gagenda[a] here *)
5.    for each inference rule "$c$ += $a_1 * a_2 * \cdots * a_k$"
6.      for $i$ from 1 to $k$
7.        for each way of instantiating the rule's variables
          such that $a_i = a$
8.          for $h$ from 1 to $k$ such that $a_h$ is not a side cond.
                (* find $\partial\mathsf{goal}/\partial agenda[c] \cdot \partial agenda[c]/\partial(a_h$ factor) *)
9.            $\gamma := \prod_{j=1}^{k} \begin{cases} gagenda[c] & \text{if } j = h \\ old & \text{if } j \neq h \text{ and } j < i \text{ and } a_j = a \\ \Delta & \text{if } j \neq h \text{ and } j = i \\ chart[a_j] & \text{otherwise} \end{cases}$
10.           if $h \neq i$ then $gchart[a_h]$ += $\gamma$
11.           if $h \leq i$ and $a_h = a$ then $\Gamma$ += $\gamma$
12.   $gagenda[a] := \Gamma$
13.   $chart[a] := old$
14. return $gagenda$[a] for each axiom $a$
```

Figure 4: An efficient algorithm for computing $\nabla\mathsf{goal}$ (even when goal is an early-stopping approximation), specialized to the case $\langle \mathbb{W}, \oplus, \otimes\rangle = \langle \mathbb{R}, +, *\rangle$. The proof is suppressed for lack of space.

time t is considered to be a different variable (possibly with different value) than the same item at time $t + 1$. The reverse pass must recover earlier values. Our somewhat tricky algorithm is shown in Fig. 4.

At line 3, a stack is needed to remember the sequence of $\langle a, old, \Delta\rangle$ triples from the original computation.[20] It is a more efficient version of the "tape" usually used in automatic differentiation. For example, it uses $O(n^2)$ rather than $O(n^3)$ space for the CKY algorithm. The trick is that Fig. 3 does not record all its computations, but only its sequence of items. Fig. 4 then re-runs the inference rules to reconstruct the computations in an acceptable order.

This method is a generalization of Eisner's (2001) prioritized forward-backward algorithm for infinite-state machines. As Eisner (2001) pointed out, the tape created on the first forward pass can also be used to speed up later passes (i.e., after the numerical optimizer has adjusted the axiom weights).[21]

[20]If one is willing to risk floating-point error, then one can store only $\langle a, old\rangle$ on the stack and recover Δ as $chart[a] - old$. Also, $agenda[a]$ and $gagenda[a]$ can be stored in the same location, as they are only used during the forward and the backward pass, respectively.

[21]In brief, a later forward pass that chooses a at Fig. 3, line 4 according to the recorded tape order (1) is faster than using a priority queue, (2) avoids ordering-related discontinuities in the objective function as the axiom weights change, (3) can prune by skipping useless updates a that scarcely affected goal (e.g.,

constituents not in any good parse) by consulting *gagenda*$[a]$ values that the previous backward pass can have written onto the tape (overwriting Δ or *old*).

5.3 Parameter estimation

To support parameter training using these gradients, our implementation of Dyna includes a training module, DynaMITE. DynaMITE supports the EM algorithm (and many variants), supervised and unsupervised training of log-linear ("maximum entropy") models using quasi-Newton methods, and smoothing-parameter tuning on development data. As an object-oriented C++ library, it also facilitates rapid implementation of new estimation techniques (Smith and Eisner, 2004; Smith and Eisner, 2005).

6 Program Transformations

Another interest of Dyna is that its high-level specifications can be manipulated by mechanical source-to-source program transformations. This makes it possible to derive new algorithms from old ones. §5.1 already sketched the *gradient* transformation for finding ∇goal. We note a few other examples.

Bounding transformations generate a new program that computes upper or lower bounds on goal, via generic bounding techniques (Prieditis, 1993; Culberson and Schaeffer, 1998). The A* heuristics explored by Klein and Manning (2003a) can be seen as resulting from bounding transformations.

With John Blatz, we are also exploring transformations that can result in asymptotically more efficient computations of goal. Their unweighted versions are well-known in the logic programming community (Tamaki and Sato, 1984; Ramakrishnan, 1991). *Folding* introduces new intermediate items, perhaps exploiting the distributive law; applications include parsing speedups such as (Eisner and Satta, 1999), as well as well-known techniques for speeding up multi-way database joins, constraint programming, or marginalization of graphical models. *Unfolding* eliminates items; it can be used to specialize a parser to a particular grammar and then to eliminate unary rules. *Magic templates* introduce top-down filtering into the search strategy and can be used to derive Earley's algorithm (Minnen, 1996), to introduce left-corner filters, and to restrict FSM constructions to build only accessible states.

Finally, there are low-level optimizations. *Term* transformations restructure terms to change their layout in memory. We are also exploring the introduction of declarations that control which items use the agenda or are memoized in the chart. This can be used to support lazy or "on-the-fly" computation (Mohri et al., 1998) and asymptotic space-saving tricks (Binder et al., 1997).

7 Usefulness of the Implementation

7.1 Applications

The current Dyna compiler has proved indispensable in our own recent projects, in the sense that we would not have attempted many of them without it.

In some cases, we were experimenting with genuinely new algorithms not supported by any existing tool, as in our work on dependency-length-limited parsing (Eisner and Smith, 2005b) and loosely syntax-based machine translation (Eisner and D. Smith, 2005). (Dyna would have been equally helpful in the first author's earlier work on new algorithms for lexicalized and CCG parsing, syntactic MT, transformational syntax, trainable parameterized FSMs, and finite-state phonology.)

In other cases (Smith and Eisner, 2004; Smith and Smith, 2004; Smith et al., 2005), Dyna let us quickly replicate, tweak, and combine useful techniques from the literature. These techniques included unweighted FS morphology, conditional random fields (Lafferty et al., 2001), synchronous parsers (Wu, 1997; Melamed, 2003), lexicalized parsers (Eisner and Satta, 1999),[22] partially supervised training à la (Pereira and Schabes, 1992),[23] and grammar induction (Klein and Manning, 2002). These replications were easy to write and extend, and to train via §5.2.

7.2 Experiments

We compared the current Dyna compiler to hand-built systems on a variety of parsing tasks. These problems were chosen not for their novelty or interesting structure, but for the availability of existing well-tuned implementations.

Best parse. We compared a Dyna CFG parser to the Java parser of Klein and Manning (2003b),[24]

[22]Markus Dreyer's reimplementation of the complex Collins (1999) parser uses under 30 lines of Dyna.

[23]For example, lines 2–3 of Fig. 1 can be extended with whenever permitted(X,I,K).

[24]Neither uses heuristics from Klein and Manning (2003a).

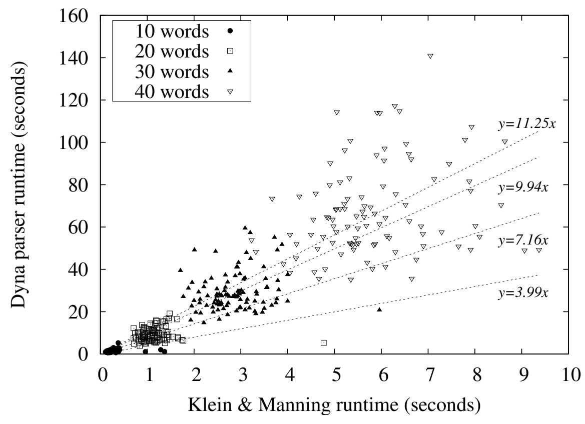


Figure 5: Dyna CKY parser vs. Klein & Manning hand-built parser, comparing runtime.

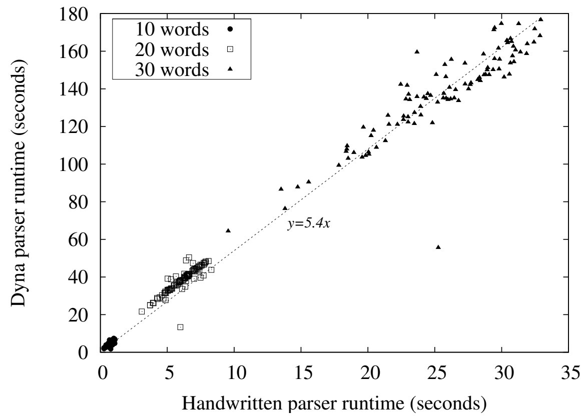


Figure 6: Dyna CKY parser vs. C++PARSE, a similar hand-built parser. The implementation differences amount to storage and indexing and give a consistent 5-fold speedup.

on the same grammar. Fig. 5 shows the results. Dyna's disadvantage is greater on longer sentences—probably because its greater memory consumption results in worse cache behavior.[25]

We also compared a Dyna CKY parser to our own hand-built implementation, C++PARSE. C++PARSE is designed like the Dyna parser but includes a few storage and indexing optimizations that Dyna does not yet have. Fig. 6 shows the 5-fold speedup from these optimizations on binarized-Treebank parsing with a large 119K-rule grammar. The sharp diagonal indicates that C++PARSE is simply a better-tuned version of the Dyna parser.

These optimizations and others are now being incorporated into the Dyna compiler, and are expected to provide similar speedups, putting Dyna's parser in the ballpark of the Klein & Manning parser. Importantly, these improvements will speed up existing Dyna programs through recompilation.

[25]Unlike Java, Dyna does not yet decide automatically when to perform garbage collection. In our experiment, garbage collection was called explicitly after each sentence and counted as part of the runtime (typically 0.25 seconds for 10-word sentences, 5 seconds for 40-word sentences).

	99%		99.99%	
uniform	89.3	(4.5)	90.3	(4.6)
after 1 EM iteration	82.9	(6.8)	85.2	(6.9)
after 2 EM iterations	77.1	(8.4)	79.1	(8.3)
after 3 EM iterations	71.6	(9.4)	73.7	(9.5)
after 4 EM iterations	66.8	(10.0)	68.8	(10.2)
after 5 iterations	62.9	(10.3)	65.0	(10.5)

Table 1: Early stopping. Each row describes a PCFG at a different stage of training; later PCFGs are sharper. The table shows the percentage of agenda runtime (mean across 1409 sentences, and standard deviation) required to get within 99% or 99.99% of the true value of goal.

Inside parsing. Johnson (2000) provides a C implementation of the inside-outside algorithm for EM training of PCFGs. We ran five iterations of EM on the WSJ10 corpus[26] using the Treebank grammar from that corpus. Dyna took 4.1 times longer.

Early stopping. An advantage of the weighted agenda discipline (§4.2) is that, with a reasonable priority function such as an item's inside probability, the inside algorithm can be stopped early with an estimate of goal's value. To measure the goodness of this early estimate, we tracked the progression of goal's value as each sentence was being parsed. In most instances, and especially after more EM iterations, the estimate was very tight long before all the weight had been accumulated (Table 1). This suggests that early stopping is a useful training speedup.

PRISM. The implemented tool most similar to Dyna that we have found is PRISM (Zhou and Sato, 2003), a probabilistic Prolog with efficient tabling and compilation. PRISM inherits expressive power from Prolog but handles only probabilities, not general semirings (or even side conditions).[27] In CKY parsing tests, PRISM was able to handle only a small fraction of the Penn Treebank ruleset (2,400 high-probability rules) and tended to crash on long sentences. Dyna is designed for real-world use: it consistently parses over 10$\times$ faster than PRISM and scales to full-sized problems.

IBAL (Pfeffer, 2001) is an elegant and powerful language for probabilistic modeling; it generalizes Bayesian networks in interesting ways.[28] Since

[26]Sentences with $\leq$10 words, stripping punctuation.

[27]Thus it can handle a subset of the cases described by Goodman (1999), again by building the whole parse forest.

[28]It might be possible to implement IBAL in Dyna (Pfeffer,

PCFGs and marginalization can be succinctly expressed in IBAL, we attempted a performance comparison on the task of the inside algorithm (Fig. 1). Unfortunately, IBAL's algorithm appears not to terminate if the PCFG contains any kind of recursion reachable from the start symbol.

8 Conclusions

Weighted deduction is a powerful theoretical formalism that encompasses many NLP algorithms (Goodman, 1999). We have given a bottom-up "inside" algorithm for general semiring-weighted deduction, based on a prioritized agenda, and a general "outside" algorithm that correctly computes weight gradients even when the inside algorithm is pruned.

We have also proposed a declarative language, Dyna, that replaces Prolog's Horn clauses with "Horn equations" over terms with values. Dyna can express more than the semiring-weighted dynamic programs treated in this paper. Our ongoing work concerns the full Dyna language, program transformations, and feedback-directed optimization.

Finally, we evaluated our first implementation of a Dyna-to-C++ compiler (download and documentation at `http://dyna.org`). We hope it will facilitate EMNLP research, just as FS toolkits have done for the FS case. It produces code that is slower than hand-crafted code but acceptably fast for our NLP research, where it has been extremely helpful.

References

J. Binder, K. Murphy, and S. Russell. 1997. Space-efficient inference in dynamic probabilistic networks. In *Proc. of IJCAI*.

S. A. Caraballo and E. Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *CL*, 24(2):275–298.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. In *Proc. of COLING-ACL*.

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

J. C. Culberson and J. Schaeffer. 1998. Pattern databases. *Computational Intelligence*.

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. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical CFGs and head-automaton grammars. In *Proc. of ACL*.

J. Eisner and D. A. Smith. 2005a. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. Technical report, Johns Hopkins U.

J. Eisner and N. A. Smith. 2005b. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*.

p.c.). Dyna is a lower-level language that itself knows nothing about the semantics of probability models, but whose inference rules could be used to implement any kind of message passing.

J. Eisner, E. Goldlust, and N. A. Smith. 2004. Dyna: A declarative language for implementing dynamic programs. In *Proc. of ACL* (companion vol.).

J. Eisner. 2001. *Smoothing a Probabilistic Lexicon via Syntactic Transformations*. Ph.D. thesis, U. of Pennsylvania.

J. Eisner. 2002. Parameter estimation for probabilistic FS transducers. In *Proc. of ACL*.

J. Goodman. 1999. Semiring parsing. *CL*, 25(4):573–605.

A. Griewank and G. Corliss, editors. 1991. *Automatic Differentiation of Algorithms*. SIAM.

M. Johnson. 2000. Inside-outside (computer program). `http://www.cog.brown.edu/~mj/Software.htm`.

L. Karttunen, J.-P. Chanod, G. Grefenstette, and A. Schiller. 1996. Regular expressions for language engineering. *JNLE*, 2(4):305–328.

M. Kifer and V. S. Subrahmanian. 1992. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12(4):335–368.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for grammar induction. In *Proc. of ACL*.

D. Klein and C. D. Manning. 2003a. A* parsing: Fast exact Viterbi parse selection. In *Proc. of HLT-NAACL*.

D. Klein and C. D. Manning. 2003b. Accurate unlexicalized parsing. In *Proc. of ACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

I. D. Melamed. 2003. Multitext grammars and synchronous parsers. In *Proc. HLT-NAACL*.

G. Minnen. 1996. Magic for filter optimization in dynamic bottom-up processing. In *Proc. of ACL*.

M. Mohri, F. Pereira, and M. Riley. 1998. A rational design for a weighted FST library. *LNCS*, 1436.

M.-J. Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *CL*, 29(1):135–143.

I. Niemelä. 1998. Logic programs with stable model semantics as a constraint programming paradigm. In *Proc. Workshop on Computational Aspects of Nonmonotonic Reasoning*.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*.

F. Pereira and D. H. D. Warren. 1983. Parsing as deduction. In *Proc. of ACL*.

A. Pfeffer. 2001. IBAL: An integrated Bayesian agent language. In *Proc. of IJCAI*.

A. Prieditis. 1993. Machine discovery of effective admissible heuristics. *Machine Learning*, 12:117–41.

R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Seshadri. 1994. The CORAL deductive system. *The VLDB Journal*, 3(2):161–210.

R. Ramakrishnan. 1991. Magic templates: a spellbinding approach to logic programs. *J. Log. Program.*, 11(3-4):189–216.

K. A. Ross and Y. Sagiv. 1992. Monotonic aggregation in deductive databases. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

S. M. Shieber, Y. Schabes, and F. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

K. Sikkel. 1997. *Parsing Schemata: A Framework for Specification and Analysis of Parsing Algorithms*. Texts in Theoretical Computer Science. Springer.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

D. A. Smith and N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*.

N. A. Smith, D. A. Smith, and R. W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proc. of HLT-EMNLP*.

G. Specht and B. Freitag. 1995. AMOS: A NL parser implemented as a deductive database in LOLA. In *Applications of Logic Databases*. Kluwer.

A. Stolcke. 1995. An efficient probabilistic CF parsing algorithm that computes prefix probabilities. *CL*, 21(2):165–201.

H. Tamaki and T. Sato. 1984. Unfold/fold transformation of logic programs. In S. Å. Tärnlund, editor, *Proceedings Second International Conference on Logic Programming*, pages 127–138, Uppsala University.

R. J. Williams and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *CL*, 23(3):377–404.

N.-F. Zhou and T. Sato. 2003. Toward a high-performance system for symbolic and statistical modeling. In *Proc. of Workshop on Learning Statistical Models from Relational Data*.

U. Zukowski and B. Freitag. 1997. The deductive database system *LOLA*. In *Logic Programming and Nonmonotonic Reasoning*, LNAI 1265. Springer.

Learning What to Talk About in Descriptive Games

Hugo Zaragoza
Microsoft Research
Cambridge, United Kingdom
hugoz@microsoft.com

Chi-Ho Li
University of Sussex
Brighton, United Kingdom
C.H.Li@sussex.ac.uk

Abstract

Text generation requires a planning module to select an object of discourse and its properties. This is specially hard in *descriptive games*, where a computer agent tries to describe some aspects of a game world. We propose to formalize this problem as a Markov Decision Process, in which an optimal message policy can be defined and learned through simulation. Furthermore, we propose *back-off policies* as a novel and effective technique to fight state dimensionality explosion in this framework.

1 Introduction

Traditionally, text generation systems are decomposed into three modules: the *application* module which manages the high-level task representation (state information, actions, goals, etc.), the *text planning* module which chooses messages based on the state of the application module, and the *sentence generation* module which transforms messages into sentences. The planning module greatly depends on the characteristics of both the application and the generation modules, solving issues in domain modelling, discourse and sentence planning, and to some degree lexical and feature selection (Cole et al., 1997). In this paper we concentrate on one of the most basic tasks that text planning needs to solve: selecting the message content, or more simply, choosing *what to talk about*.

Work on text-generation often assumes that an object or topic has been already chosen for discussion. This is reasonable for many applications, but in some cases choosing *what* to talk about can be harder than choosing *how* to. This is the case in the type of text generation applications that we are interested in: generating descriptive messages in computer games. In a modern computer game at any given moment there may be an enormous number of object properties that can be described, each with varying importance and consequences. The outcome of the game depends not only on the skill of the player, but also on the quality of the descriptive messages produced. We refer to such situations as *descriptive games*.

Our goal is to develop a strategy to choose the most interesting descriptive messages that a particular talker may communicate to a particular listener, given their context (i.e. their knowledge of the world and of each-other). We refer to this as *message planning*.

Developing a general framework for planning is very difficult because of the strong coupling between the planning and application modules. We propose to frame message planning as a Markov Decision Process (MDP) which encodes the environment, the information available to the *talker* and *listener*, the consequences of their communicative and non-communicative acts, and the constraints of the text generation module. Furthermore we propose to use Reinforcement Learning (RL) to learn the optimal message policy. We demonstrate the overall principle (Section 2) and then develop in more detail a computer game setting (Section 3).

One of the main weaknesses of RL is the problem of state dimensionality explosion. This problem is specially acute in message planning, since in typical situations there can be hundreds of thousands of potential messages. At the same time, the domain is highly structured. We propose to exploit this structure using a form of the back-off smoothing principle on the state space (Section 4).

1.1 Related Work

Our problem setting can be seen as a generalisation of the *content selection* problem in the generation of referring expressions in NLG. In the standard setting of this problem (see for example (van Deemter and Krahmer, to appear)) an algorithm needs to select the *distinguishing description* of an object in a scene. This description can be seen as a subset of scene properties which i) uniquely identifies a given target object, and ii) is optimal in some sense (minimal, psychologically plausible, etc.) van Deemter and Krahmer show that most content selection algorithms can be described as different cost functions over a particular graph representation of the scene. Minimising the cost of a subgraph leads to a distinguishing description.

Some aspects of our work generalise that of content selection: i) we consider the target object is unknown, ii) we consider scenes (i.e. world states) that are *dynamic* (i.e. they change over time) and *reactive* (i.e. utterances change the world), and iii) we consider listeners that have partial knowledge of the scene. This has important consequences. For example, the cost of a description cannot be directly evaluated; instead, we must *play the game*, that is, generate utterances and observe the rewards obtained over time. Also identical word-states may lead to different optimal messages, depending on the listener's partial knowledge. Other aspects of our work are very simplistic compared to current work in content selection, for example with respect to the use of negation and of properties that are boolean, relative or graded (van Deemter and Krahmer, to appear). We hope to incorporate these ideas into our work soon.

Probabilistic dialogue policies have been previously proposed for spoken dialogue systems (SDS) (see for example (Singh et al., 2002; Williams et al., 2005) and references therein). However, work in SDS focus mainly on coping with the noise and uncertainty resulting from speech recognition and sentence parsing. In this context MDPs are used to infer features and plan communicative strategies (modality, confusion, initiative, etc.) In our work we do not need to deal with uncertainty or parsing; our main concern is in the selection of the message content. In this sense our work is closer to (Henderson et al., 2005), where RL is used to train a SDS with very many states encoding message content.

Finally, with respect to the state-explosion problem in RL, related work can be found in the areas of multi-task learning and robot motion planning (Dietterich, 2000, and references therein). In these works the main concern is identifying the features that are relevant to specific sub-tasks, so that robots may learn multiple loosely-coupled tasks without incurring state-explosion. (Henderson et al., 2005) also addresses this problem in the context of SDS and proposes a semi-supervised solution. Our approach is related to these works, but it is different in that we assume that the feature structure is known in advance and has a very particular form amenable to a form of back-off regularisation.

2 Message planning

Let us consider an environment comprising a world with some objects and some agents, and some dynamics that govern their interaction. Agents can observe and memorize certain things about the world, can carry out actions and communicate with other agents. As they do so, they are rewarded or punished by the environment (e.g. if they find food, if the complete some goal, if they run out of energy, etc.)

The agents' actions are governed by a *policy*. We will consider separately the *physical action policy* (π), which decides which physical action to take given the state of the agent, and the *message action policy* (μ), which decides when to communicate, to whom, and what about. Our main concern in this paper will be to learn an optimal μ. Before we define this goal more precisely, we will introduce some notation.

A *property* is a set of *attribute-value pairs*. An *object* is a set of properties, with (at least) attributes Type and Location. A *domain* is a set of objects. Furthermore, we say that s' is a *sub-domain* of s if s' can be obtained by deleting property–value pairs from s (while enforcing the condition that remaining objects must have Type and Location). $\mathsf{Sub}(s)$ is the set containing s, all sub-domains of s, and the empty domain $\emptyset$.

A *world state* can be represented as a domain, noted s_W. Any partial view of the world state can also be represented as a domain $s \in \mathsf{Sub}(s_W)$. Similarly the content of any descriptive message about the world, noted m, can be represented as a partial view of it. An *agent* is the tuple:

$$A := \Big(s_A, \pi_A, \{\mu_{AA'}, s_{AA'}\}_{A' \neq A}\Big)$$

- $s_A \in \mathsf{Sub}(s_W)$: knowledge that A has about the state of the world.
- $s_{AA'} \in \mathsf{Sub}(s_A \cap s'_A)$: knowledge that A has about the knowledge that A' has about the world.
- $\pi_a := P(c|s_A)$ is the action policy of A, and c is a physical action.
- $\mu_{AA'} := P(m \in \mathcal{M}(s_A)|s_A, s_{AA'})$ is the message policy of A for sending messages to A', and $\mathcal{M}(s_A)$ are all valid messages at state s_A (discussed in Section 2.3).

When an agent A decides to send a message to A', it can use its knowledge of A' to choose messages effectively. For example, A will prefer to describe things that it knows A' does not know (i.e. not in $s_{AA'}$). This is the reason why the message policy μ_A depends on both s_A and $s_{AA'}$. After a message is sent (i.e. realised and uttered) the agent's will update their knowledge states $s_{A'}$, $s_{A'A}$ and $s_{AA'}$.

The question that we address in this paper is that of *learning an optimal message policy* $\mu_{AA'}$.

2.1 Talker's Markov Decision Process

We are going to formalize this problem as a standard Markov Decision Process (MDP). In general a MDP (Sutton and Barto, 1998) is defined over some set of states $\mathcal{S} := \{s_i\}_{i=1..K}$ and actions associated to every state, $\mathcal{A}(s_i) := \{a_{ij}\}_{j=1..N_i}$. The environment is governed by the state transition function $\mathcal{P}^a_{ss'} := P(s'|s,a)$. A policy determines the likelihood of actions at a given state: $\pi(s) := P(a|s)$. At each state transition a reward is generated from the reward function $\mathcal{R}^a_{ss'} := E\{r|s,s',a\}$.

MDPs allow us to define and find *optimal policies* which maximise the expected reward. Classical MDPs assume that the different functions introduced above are known and have some tractable analytical form. Reinforcement Learning (RL) in as extension of MDPs in which the environment function $\mathcal{P}^a_{ss'}$ is unknown or complex, and so the optimal policy needs to be learned online by directly interacting with the environment. There exist a number of algorithms to solve a RL problem, such as Q-Learning or SARSA (Sutton and Barto, 1998).

We can use a MDP to describe a full descriptive game, in which several agents interact with the world and communicate with each-other. To do so we would need to consider composite states containing s_W, $\{s_A\}_A$, and $\Big\{\{s_{AA'}\}_{A' \neq A}\Big\}_A$. Similarly, we need to consider composite policies containing $\{\pi_A\}_A$ and $\Big\{(\mu_{AA'})_{A' \neq A}\Big\}_A$. Finally, we would consider the many constrains in this model; for example: only physical actions affect the state of the world, only message actions affect believes, and only believe states can affect the choice of the agent's actions.

MDPs provide us with a principled way to deal with these elements and their relationships. However, dealing with the most general case results in models that are very cumbersome and which hide the conceptual simplicity of our approach. For this reason, we will limit ourselves in this paper to one of the simplest communication cases of interest: a single all-knowing talker, and a single listener completely observed by the talker. We will discuss later how this can be generalized.

2.2 The *Talking God* Setting

In the simplest case, an all-knowing agent A_0 sits in the background, without taking any physical actions, and uses its message policy (μ_{01}) to send messages to a *listener* agent A_1. The listener agent cannot talk back, but can interact with the environment using its physical action policy π_1. Rewards obtained by A_1 are shared by both agents. We refer to this setting as the *talking God* setting. Examples of such situations are common in games, for example when a computer character talks to its (computer) team-

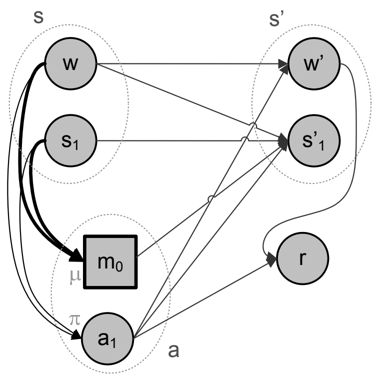


Figure 1: *Talking God* MDP.

mates, or when a mother-ship with full information of the ground sends a small simple robot to do a task. Another example would be that of a teacher talking to a learner, except that the teacher may not have full information of the learners head!

Since the talker is all-knowing, it follows that $s_0 = s_W$ and $s_{01} = s_1$. Furthermore, since the talker does not take physical actions, π_0 does not need to be defined. Similarly, since the listener does not talk we do not need to define μ_{10} or s_{10}. This case is depicted in Figure 1 as a graphical model. By grouping states and actions (dotted lines) we can see that this is can be modelled as a standard MDP. If all the probability distributions are known analytically, or if they can be sampled, optimal physical and message policies can be learnt (thick arrows).

Several generalizations of this model are possible. A straight forward generalization is to consider more than one listener agent. We can then choose to learn a single policy for all, or individual policies for each agent.

A second way to generalize the setting is to make the listeners mind only partially observable to the talker. In this case the talker continues to know the entire world ($s_0 = s_W$), but does not know exactly what the listener knows ($s_{01} \neq s_0$). This is more realistic in situations in which the listener cannot *talk back* to the talker, or in which the talkers mind is not observable. However, to model this we need a partially observable MDP (POMDP). Solving POMDPS is much harder than solving MDPs, but there have been models proposed for dialogue management (Williams et al., 2005).

In the more general case, the talker would have partial knowledge of the world and of the listener, and would itself act. In that case all agents are equal and can communicate as they evolve in the environment. The other agents minds are not directly observable, but we obtain information about them from their actions and their messages. This can all be in principle modelled by POMDPs in a straightforward manner, although solving these models is more involved. We are currently working towards doing so.

Finally, we note that all the above cases have dealt with worlds in which objects are static (i.e. information does not become obsolete), agents do not gain or communicate erroneous information, and communication itself is non-ambiguous and loss-less. This is a realistic scenario for text generation, and for communication between computer agents in games, but it is far removed from the spoken dialogue setting.

2.3 Generation Module and Valid Messages

Generating descriptive sentences of domains can be done in a number of ways, from template to feature-based systems (Cole et al., 1997). Our framework does not depend on a particular choice of generation module, and so we do not need to discuss this module. However, our message policy is not decoupled of the generation module; indeed, it would not make sense to develop a planning module which plans messages that cannot be realised! In our framework, the generation module is seen simply as a fixed and known *filter* over all possible the messages.

We formalize this by representing an agent's generation module as a function $\Gamma_A(m)$ mapping a message m to a NL sentence, or to $\emptyset$ if the module cannot fully realise m. The set of available messages to an agent A in state s_A is therefore: $\mathcal{M}(s_A) := \{m \,|\, m \in \mathsf{Sub}(s_A)\,,\ \Gamma_A(m) \neq \emptyset\}$.

3 A Simple Game Example

In this section we will use a simple computer game to demonstrate how the proposed framework can be used to learn message policies.

The game evolves in a grid-world. A mother-ship sends a *scout*, which will try to move from its

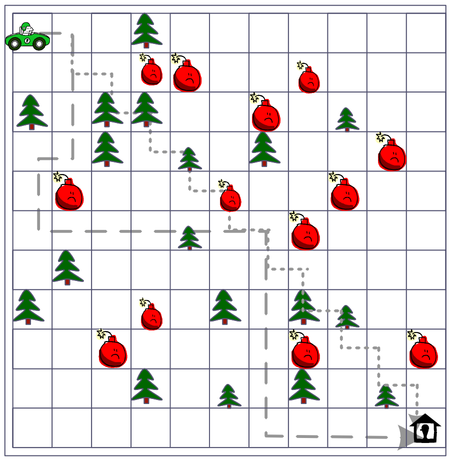


Figure 2: Example of a Simple Game Board.

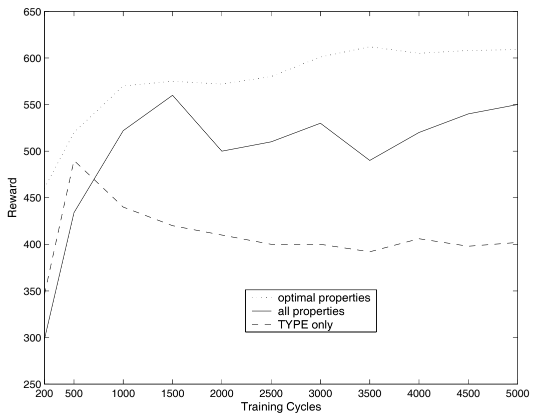


Figure 3: Simple Game Learning Results

State	**Best Action Learnt** *(and possible sentence realisation)*
{ TREE-BIG-LEFT }	∅ *-SILENCE-*
{ BOMB-BIG-FRONT }	BOMB-FRONT *There is a bomb in front of you*
{ TREE-SMALL-LEFT, TREE-BIG-RIGHT }	TREE-BIG-RIGHT *There is a big tree to your right*
{ BOMB-BIG-FRONT, BOMB-SMALL-LEFT, TREE-BIG-RIGHT, TREE-SMALL-BACK }	TREE-BIG-RIGHT *There is a big tree to your right*

Table 1: Examples of learnt actions.

starting position (top left corner) to a target (bottom right). There are two types of objects on the board, Type := {bomb, tree}, with a property Size := {big, small} in addition of Location. If a scout attempts to move into a big tree, the move is blocked; small trees have no effect. If a scout moves into a bomb the scout is destroyed and a new one is created at the starting position. Before every step the mother-ship may send a message to the scout. Then the scout moves one step (horizontal or vertical) towards the target choosing the shortest path which avoids hazards known by the scout (the A* algorithm is used for this). Initially scouts have no knowledge of the objects in the world; they gain this knowledge by stepping into objects or by receiving information from the mother-ship.

This is an instance of the talking god model discussed previously. The scout is the listener agent (A_1), and the mother-ship the talker (A_0). The scouts action policy π_1 is fixed (as described above), but we need to learn the message policy μ_{01}.

Rewards are associated with the results of physical actions: a high positive reward (1000) is assigned to reaching the destination, a large negative reward (-100) to stepping in a bomb, a medium negative reward (-10) to being blocked by a big tree, a small negative reward to every step (-1). Furthermore, sending a message has a small negative reward proportional to the number of attributes mentioned in the message (-2 per attribute, to discourage the talker from sending useless information). The message $\emptyset$ is given zero cost; this is done in order to learn when *not* to talk.

Learning is done as follows. We designed five maps of 11×11 cells, each with approximately 15 bombs and 20 trees of varying sizes placed in strategic locations to make the scouts task difficult (one of these maps is depicted in Figure 2; an A* path without any knowledge and one with full knowledge of the board are shown as dotted and dashed arrows respectively). A training epoch consists of randomly drawing one of these maps and running a single game until completion. The SARSA algorithm is used to learn the message policy, with $\epsilon = 0.1$ and $\gamma = 0.9$. The states s_W and s_1 are encoded to represent the location of objects surrounding the scout, relative to its direction (i.e. objects directly in front of the agent always receive the same location value). To speed up training, we only consider the 8 cells adjacent to the agent.

Figure 3 shows the results of these experiments. For comparison, we note that completing the game with a uniformly random talking policy results in an average reward of less than -3000 meaning that on average more than 30 scouts die before the target is reached. The dashed line indicates the reward obtained during training for a policy which does not use the size attribute, but only type and location. This policy effectively learns that both bombs and trees in front of the agent are to be communicated, resulting in an average reward of approximately 400, and reducing the average number of deaths to less than 2. The solid line represents the results obtained by a policy that is forced to use all attributes. Despite the increase in communication cost, this policy can distinguish between small and large trees, and so it increases the overall reward two-fold. Finally, the dotted line represents the results obtained by a policy that can choose whether to use or not the size attribute. This policy proves to be even more effective than the previous one; this means that it has learnt to use the size attribute only when it is necessary. Some optimal (state,action) pairs learnt for this policy are shown in Table 1. The first three show correctly learnt optimal actions. The last is an example of a wrongly learnt action, due to the state being rare.

These are encouraging results, since they demonstrate in practice how optimal policies may be learnt for message planning. However, it should be clear form this example that, as we increase the number of types, attributes and values, this approach will become unfeasible. This is discussed in the next section.

4 Back-Off Policies

One of the main problems when using RL in practical settings (and, more generally, using MDPs) is the exponential growth of the state space, and consequently of the learning time required. In our case, if there are M attributes, and each attribute p_i has $N(p_i)$ values, then there are $S = \prod_{i=1}^{M} N(p_i)$ possible sub-domains, and up to 2^S states in the state space. This exponential growth, unless addressed, will render MDP learning unfeasible.

NL domains are usually rich with structure, some of it which is known *a priori*. This is the case in text generation of descriptions for computer games, where we have many sources of information about the objects of discourse (i.e. world ontology, dynamics, etc.) We propose to tackle the problem of state dimensionality explosion by using this structure explicitly in the design of hierarchical policies.

We do so by borrowing the back-off smoothing idea from language models. This idea can be stated as: train a set of probability models, ordered by their specificity, and make predictions using the most specific model possible, but only if there is enough training data to support its prediction; otherwise, *back-off* to the next less-specific model available.

Formally, let us assume that for every state s we can construct a sequence of K embedded partial representations of increasing complexity, $(s_{[1]}, \ldots, s_{[k]}, \ldots, s_{[K]})$. Let us denote $\hat{\pi}_{[k]}$ a sequence of policies operating at each of the partial representation levels respectively, and let each of these policies have a *confidence measurement* $c_k(s)$ indicating the quality of the prediction at each state. Since k indicates increasingly complex, we require that $c_k(s) \geq c_{k'}(s)$ if $k < k'$. Then, the most specific policy we can use at state s can be written as:

$$k^*_s := \arg\max_k \{k \cdot \mathrm{sign}\,(c_k(s) - \theta)\} \qquad (1)$$

A back-off policy can be implemented by choosing, at every state s the most specific policy available:

$$\pi(s) = \hat{\pi}_{[k^*_s]}(s_{[k^*_s]}) \qquad (2)$$

We can use a standard off-policy learning algorithm (such as Q-learning or SARSA) to learn all the policies simultaneously. At every step, we draw an action using (2) and update all policies with the obtained reward[1]. Initially, the learning will be driven by high-level (simple) policies. More complex policies will kick-in progressively for those states that are encountered more often.

In order to implement back-off policies for our setting, we need to define a confidence function c_k. A simple confidence measure is the number of times the state $s_{[k]}$ has been previously encountered. This measure grows on average very quickly for small k states and slowly for high k states. Nevertheless, reoccurring similar states will have high visit counts

[1]An alternative view of back-off policies is to consider that a single complete policy is being learnt, but that actions are being drawn from regularised versions of this policy, where the regularisation is a back-off model on the features. We show this in Appendix I

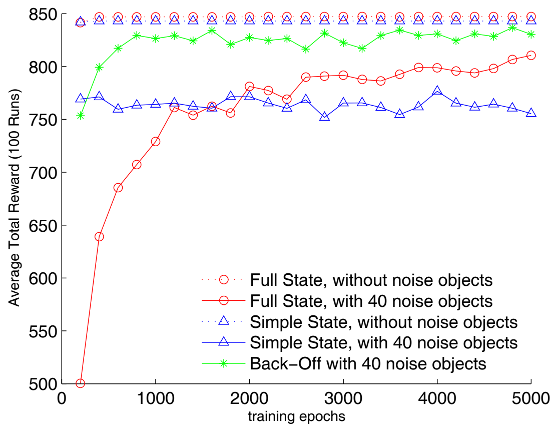


Figure 4: Back-Off Policy Simulation Results.

for all k values. This is exactly the kind of behaviour we require.

Furthermore, we need to choose a set of representations of increasing complexity. For example, in the case of n-gram models it is natural to choose as representations sequences of preceding words of increasing size. There are many choices open to us in our application domain. A natural choice is to order attribute types by their importance to the task. For example, at the simplest level of representation objects can be represented only by their type, at a second level by the type and colour, and at a third level by all the attributes. This same technique could be used to exploit ontologies and other sources of knowledge. Another way to create levels of representation of increasing detail is to consider different perceptual windows. For example, at the simplest level the agent can consider only objects directly in front of it, since these are generally the most important when navigating. At a second level we may consider also what is to the left and right of us, and finally consider all surrounding cells. This could be pursued even further by considering regions of increasing size.

4.1 Simulation Results

We present here a series of experiments based on the previous game setting, but further simplified to pinpoint the effect of dimensionality explosion, and how back-off policies can be used to mitigate it.

We modify the simple game of Section 3 as follows. First, we add a new object type, stone, and a new property Colour := {red, green}. We let all trees be green and big and all bombs red and small, and furthermore we fix their location (i.e. we use one map instead of five). Finally we change the world behaviour so that an agent that steps into a bomb receives the negative reward but does not die, it continues until it reaches the target. All these changes are done to reduce the variability of our learning baseline.

At every game we generate 40 stones of random location, size and colour. Stepping on stones has no physical effect to the scout and it generates the same reward as moving into an empty cell, but this is unknown to the talker and will need to be learnt. These stones are used as *noise* objects, which increase the size of the state space. When there are no noise objects, the number of possible states is $3^8 \approx 6.5K$ (the actual number of states will be much smaller since there is a single maze). Noise objects can take $2 \times 2 = 4$ possible forms, so the total number of states with noise objects is $(3+4)^8 \approx 6M$. Even with such a simplistic example we can see how drastic the state dimensionality problem is. Despite the fact that the noise objects do not affect the reward structure of our simple game, reinforcement learning will be drastically slowed down by them.

Simulation results[2] are shown in Figure 4. First let us look at the results obtained using the full state representation used in Section 3 (noted Full State). Solid and dotted lines represent runs obtained with and without noise objects. First note that learning without noise objects (dotted circles) occurs mostly within the first few epochs and settles after 250 epochs. When noise objects are added (solid circles) learning greatly slows down, taking over 5K epochs. This is a typical illustration of the effect that the number of states has on the speed of learning.

An obvious way to limit the number of states is to eliminate features. For comparison, we learned a simple representation policy with states encoding only the type of the object *directly in front* of the agent, ignoring its colour and all other locations (noted Simple State). Without noise, the performance (dotted triangles) is only slightly worse than that of the original policy. However, when noise objects

[2]Every 200 training epochs we run 100 validation epochs with $\epsilon = 0$. Only the average validation rewards are plotted.

are added (solid triangles) the training is no longer slowed down. In fact, with noise objects this policy outperforms the original policy up to epoch 1000: the performance lost in the representation is made up by the speed of learning.

We set up a back-off policy with $K = 3$ as follows. We use the Simple representation at $k = 1$, plus a second level of representation where we represent the colour as well as the type of the object in front of the agent, and finally the Full representation as the third level. As the c_k function we use state visit counts as discussed above and we set $\theta = 10$. Before reaching the full policy (level 3), this policy should progressively learn to avoid bombs and trees directly in front (level 1), then (level 2) not avoid small trees directly in front. We plot the performance of this back-off policy (stars) in Figure 4. We see that it attains very quickly the performance of the simple policy (in less than 200 epochs), but the continues to increase in performance settling within 500 epochs with a performance superior to that of the full state representation, and very close to that of the policies operating in the noiseless world.

Despite the small scale of this study, our results clearly suggest that back-off policies can be used effectively to control state dimensionality explosion when we have strong prior knowledge of the structure of the state space. Furthermore (and this may be very important in real applications such as game development) we find that back-off policies produce a *natural* to feel to the errors incurred while learning, since policies develop progressively in their complexity.

5 Conclusion

We have developed a formalism to learn interactively the most informative message content given the state of the listener and the world. We formalised this problem as a MDP and shown how RL may be used to learn message policies even when the environment dynamics are unknown. Finally, we have shown the importance of tackling the problem of state dimensionality explosion, and we have proposed one method to do so which exploits explicit *a priori* ontological knowledge of the task.

References

R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue. 1997. *Survey of the State of the Art in Human Language Technology*. Cambridge University Press.

T. G. Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

J. Henderson, O. Lemon, and K. Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

S. Singh, D. Litmanand, M. Kearns, and M. Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

R. S. Sutton and A. G. Barto. 1998. *Reinforcement Learning*. MIT Press.

K. van Deemter and E. Krahmer. (to appear). Graphs and booleans. In *Computing Meaning*, volume 3 of *Studies in Linguistics and Philosophy*. Kluwer Academic Publishers.

J. D. Williams, P. Poupart, and S. Young. 2005. Factored partially observable markov decision processes for dialogue management. In *4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

6 Appendix I

We show here that the expected reward for a partial policy π_k after an action a, noted $Q^{\pi_k}(s,a)$, can be obtained from the expected reward of the full policy $Q^{\pi}(s,a)$ and the conditional state probabilities $P(s|s_{[k]})$. We may use this to compute the expected risk of any partial policy $R^{\pi_k}(s)$ from the full policy.

Let $\mathcal{T}_k(s) := \left\{ s' \in \mathcal{S} \mid s'_{[k]} = s_{[k]} \right\}$ be the subset of full states which map to the same value of s. Given a state distribution $P(s)$ we can define distributions over partial states:

$$P(s_{[k]}, s_{[j]}) = \sum_{s' \in \mathcal{T}_k(s) \cap \mathcal{T}_j(s)} P(s') \quad . \tag{3}$$

Since $\sum_{s' \in \mathcal{T}_k(s)} P(s'|s_{[k]}) = 1$, we have $P(A|s_{[k]}) = \sum_{s' \in \mathcal{T}_k(s)} P(A|s')P(s'|s_{[k]})$, and so:

$$Q^{\pi_k}(s,a) = \sum_{s' \in \mathcal{T}_k(s)} P(s'|s_{[k]}) Q^{\pi}(s',a) \quad . \tag{4}$$

Using Question Series To Evaluate Question Answering System Effectiveness

Ellen M. Voorhees
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

The original motivation for using question series in the TREC 2004 question answering track was the desire to model aspects of dialogue processing in an evaluation task that included different question types. The structure introduced by the series also proved to have an important additional benefit: the series is at an appropriate level of granularity for aggregating scores for an effective evaluation. The series is small enough to be meaningful at the task level since it represents a single user interaction, yet it is large enough to avoid the highly skewed score distributions exhibited by single questions. An analysis of the reliability of the per-series evaluation shows the evaluation is stable for differences in scores seen in the track.

The development of question answering technology in recent years has been driven by tasks defined in community-wide evaluations such as TREC, NTCIR, and CLEF. The TREC question answering (QA) track started in 1999, with the first several editions of the track focused on factoid questions. A factoid question is a fact-based, short answer question such as *How many calories are there in a Big Mac?*. The track has evolved by increasing the type and difficulty of questions that are included in the test set. The task in the TREC 2003 QA track was a combined task that contained list and definition questions in addition to factoid questions (Voorhees, 2004). A list question asks for different instances of a particular kind of information to be returned, such as *List the names of chewing gums*. Answering such questions requires a system to assemble an answer from information located in multiple documents. A definition question asks for interesting information about a particular person or thing such as *Who is Vlad the Impaler?* or *What is a golden parachute?*. Definition questions also require systems to locate information in multiple documents, but in this case the information of interest is much less crisply delineated.

Like the NTCIR4 QACIAD challenge (Kato et al., 2004), the TREC 2004 QA track grouped questions into series, using the series as abstractions of information-seeking dialogues. In addition to modeling a real user task, the series are a step toward incorporating context-processing into QA evaluation since earlier questions in a series provide some context for the current question. In the case of the TREC series, each series contained factoid and list questions and had the target of a definition associated with it. Each question in a series asked for some information about the target. In addition, the final question in each series was an explicit "other" question, which was to be interpreted as "Tell me other interesting things about this target I don't know enough to ask directly". This last question was roughly equivalent to the definition questions in the TREC 2003 task.

This paper examines the efficacy of series-based QA evaluation, and demonstrates that aggregating scores over individual series provides a more meaningful evaluation than averages of individual question scores. The next section describes the question series that formed the basis of the TREC 2004 evaluation. Since TREC uses different evaluation protocols for different question types, the following section describes the way in which individual question types were evaluated. Section 3 contrasts the scores obtained by aggregating individual question scores by question type or by series, and shows the use of series leads to a reliable evaluation at differences in scores that are observed in practice.

1 Question Series

A question series as used in the TREC 2004 QA track consisted of several factoid questions, zero to two list questions, and exactly one Other question. Associated with each series was a definition target. The series a question belonged to, the order of the question in the series, and the type of each question (factoid, list, or Other) were all explicitly encoded in the XML format used to describe the test set. Example series (minus the XML tags) are shown in figure 1. A target was a person, an organization, or thing that was a plausible match for the scenario assumed for the task: that the questioner was an "average" adult reader of US newspapers who was looking for more information about a term encountered while reading the paper.

The TREC 2004 test set contains 65 series. Of the 65 targets, 23 are PERSONs, 25 are ORGANIZATIONs, and 17 are THINGs. The series contain a total of 230 factoid questions, 56 list questions, and 65 (one per target) Other questions. Each series contains at least four questions, counting the Other question, with most series containing five or six questions. The maximum number of questions in a series is ten.

Question series were also the fundamental structure used in the QACIAD challenge (Question Answering Challenge for Information Access Dialogue) of NTCIR4. However, there are some important differences between the QACIAD and TREC series. The QACIAD series model a more natural flow of questions in an information-seeking dialogue. Given other evaluation requirements (most questions need to have an answer in the source documents, answers to earlier questions should not be given in later questions, etc.), the series in the TREC test set are heavily edited versions of the series collected from the original information seekers. The resulting edited series appear as a stilted conversational style when viewed from the perspective of true dialogue, and the series do not reflect the full range of information requested in the original series. (For example, TREC requires list question answers to be concrete entities such as cities or book titles while the information seekers often asked for fuzzier information such as lists of descriptive qualities.) The QACIAD challenge contained two types of series, gathering series and browsing series. In a gathering series, all of the questions are related to a single target (that was not explicitly given in QACIAD), while questions in a browsing series can refer to unrelated targets. The TREC series are all gathering type series with the target explicitly given. Finally, the QACIAD series consist of list questions only, since factoid questions are treated as list questions with a single answer.

Systems participating in the TREC evaluation were required to process series independently from one another, and were required to process an individual series in question order. That is, systems were allowed to use questions and answers from earlier questions in a series to answer later questions in that same series, but could not "look ahead" and use later questions to help answer earlier questions. The series was the unit used to structure the test set, but there was no requirement for systems to process a series as a unit. Some systems appended the target to each of the questions in its series and then processed all resulting question strings independently as in earlier TREC evaluations. Per-series evaluation is still valid since the task to be evaluated is defined in terms of the series and is independent of how systems choose to process the questions.

Sixty-three runs from 28 participants were submitted to the TREC 2004 QA track.

2 Scoring Question Series

The evaluation protocol for individual questions depends on the type of the question. This section summarizes the protocols for the individual question types and for a series as a whole.

3	Hale Bopp comet		
	3.1	FACTOID	When was the comet discovered?
	3.2	FACTOID	How often does it approach the earth?
	3.3	LIST	In what countries was the comet visible on its last return?
	3.4	OTHER	
21	Club Med		
	21.1	FACTOID	How many Club Med vacation spots are there worldwide?
	21.2	LIST	List the spots in the United States.
	21.3	FACTOID	Where is an adults-only Club Med?
	21.4	OTHER	
22	Franz Kafka		
	22.1	FACTOID	Where was Franz Kafka born?
	22.2	FACTOID	When was he born?
	22.3	FACTOID	What is his ethnic background?
	22.4	LIST	What books did he author?
	22.5	OTHER	

Figure 1: Sample question series from the test set. Series 3 has a THING as a target, series 21 has an ORGANIZATION as a target, and series 22 has a PERSON as a target.

2.1 Factoid questions

The system response for a factoid question is either exactly one [*doc-id*, *answer-string*] pair or the literal string 'NIL'. NIL is returned by a system when it believes there is no answer to the question in the document collection. Otherwise, *answer-string* is a string containing precisely an answer to the question, and *doc-id* is the id of a document in the collection that supports *answer-string* as an answer.

Each response was assigned exactly one of the following four judgments:

incorrect: the answer string does not contain a right answer or the answer is not responsive;

not supported: the answer string contains a right answer but the document returned does not support that answer;

not exact: the answer string contains a right answer and the document supports that answer, but the string contains more than just the answer or is missing bits of the answer;

correct: the answer string consists of exactly the right answer and that answer is supported by the document returned.

To be responsive, an answer string is required to contain appropriate units and to refer to the correct "famous" entity (e.g., the Taj Mahal casino is not responsive when the question asks about "the Taj Mahal"). NIL responses are correct only if there is no known answer to the question in the collection and are incorrect otherwise. NIL is correct for 22 of the 230 factoid questions in the test set

An individual factoid question has a binary score, 1 if the response is judged correct and 0 otherwise. The score for a set of factoid questions is accuracy, the fraction of questions in the set judged correct.

2.2 List questions

A list question can be thought of as a shorthand for asking the same factoid question multiple times. The set of all correct, distinct answers in the document collection that satisfy the factoid question is the correct answer for a list question.

A system's response for a list question is an unordered set of [*doc-id*, *answer-string*] pairs such that each *answer-string* is considered an instance of the requested type. Judgments of incorrect, unsupported, not exact, and correct are made for individual response pairs as in the factoid judging. The assessor is given one run's entire list at a time, and while judging for correctness also marks a set of responses as distinct. The assessor chooses an arbitrary member of the equivalent responses to be marked distinct, and the remainder are not marked as distinct. Only correct responses may be marked as distinct.

The final correct answer set for a list question is compiled from the union of the correct responses across all runs plus the instances the assessor found during question development. For the 55 list questions used in the evaluation (one list question was dropped because the assessor decided there were no correct answers during judging), the average number of answers per question is 8.8, with 2 as the smallest number of answers, and 41 as the maximum number of answers. A system's response to a list question was scored using instance precision (IP) and instance recall (IR) based on the list of known instances. Let S be the number of known instances, D be the number of correct, distinct responses returned by the system, and N be the total number of responses returned by the system. Then $IP = D/N$ and $IR = D/S$. Precision and recall were then combined using the F measure with equal weight given to recall and precision:

$$F = \frac{2 \times IP \times IR}{IP + IR}$$

The score for a set of list questions is the mean of the individual questions' F scores.

2.3 Other questions

The Other questions were evaluated using the same methodology as the TREC 2003 definition questions (Voorhees, 2003). A system's response for an Other question is an unordered set of [*doc-id*, *answer-string*] pairs as for list questions. Each string is presumed to be a facet in the definition of the series' target that had not yet been covered by earlier questions in the series. The requirement to not repeat information already covered by earlier questions in the series made answering Other questions somewhat more difficult than answering TREC 2003 definition questions.

Judging the quality of the systems' responses is done in two steps. In the first step, all of the answer strings from all of the systems' responses are presented to the assessor in a single list. Using these responses and the searches done during question development, the assessor creates a list of information nuggets about the target. An information nugget is an atomic piece of information about the target that is interesting (in the assessor's opinion) and is not part of an earlier question in the series or an answer to an earlier question in the series. An information nugget is atomic if the assessor can make a binary decision as to whether the nugget appears in a response. Once the nugget list is created for a target, the assessor marks some nuggets as vital, meaning that this information must be returned for a response to be good. Non-vital nuggets act as don't care conditions in that the assessor believes the information in the nugget to be interesting enough that returning the information is acceptable in, but not necessary for, a good response.

In the second step of judging the responses, an assessor goes through each system's response in turn and marks which nuggets appear in the response. A response contains a nugget if there is a conceptual match between the response and the nugget; that is, the match is independent of the particular wording used in either the nugget or the response. A nugget match is marked at most once per response—if the response contains more than one match for a nugget, an arbitrary match is marked and the remainder are left unmarked. A single [*doc-id*, *answer-string*] pair in a system response may match 0, 1, or multiple nuggets.

Given the nugget list and the set of nuggets matched in a system's response, the nugget recall of a response is the ratio of the number of matched nuggets to the total number of vital nuggets in the list. Nugget precision is much more difficult to compute since there is no effective way of enumerating all the concepts in a response. Instead, a measure based on length (in non-white space characters) is used as an approximation to nugget precision. The length-based measure starts with an initial allowance of 100 characters for each (vital or non-vital) nugget matched. If the total system response is less than this number of characters, the value of the measure is 1.0. Otherwise, the measure's value decreases as the length increases using the function $1 - \frac{length - allowance}{length}$. The final score for an Other question is computed as the F measure with nugget recall three times as important as nugget precision:

$$F(\beta = 3) = \frac{10 \times \text{precision} \times \text{recall}}{9 \times \text{precision} + \text{recall}}.$$

Note that the Other question for series S7 was mistakenly left unjudged, so the series was was removed from the TREC 2004 evaluation. This means final scores for runs were computed over 64 rather than 65 question series.

2.4 Per-series scores

In the TREC 2003 evaluation, the final score for a run was computed as a weighted average of the mean scores for different question types:

$$\begin{aligned}\text{FinalScore} &= .5\text{FactoidAccuracy} + .25\text{ListAveF}\\ &+ .25\text{DefinitionAveF}.\end{aligned}$$

Since each of the component scores ranges between 0 and 1, the final score is also in that range. The weights for the different components reflect the desire to emphasize factoid scores, since factoid technology is the most mature, while still allowing other components to affect the final score. The specific weights used match this general objective, but are otherwise arbitrary. No experiments have been run examining the effect of different weights on the stability of the final scores, but small perturbations in the weights should have little effect on the results.

An individual question series also contains a mixture of different question types, so the weighted average can be computed for an individual series rather than the test set as a whole. The mean of the per-series scores is then used as the final score for a run.

We use the same weighted average as above to compute the score for an individual series that contains all three question types, using only the scores for questions belonging to that series in the computation and using the Other question's score in place of the average of definition questions scores. For those series that did not contain any list questions, the score was computed as .67FactoidAccuracy + .33OtherF. Figure 2 shows the average per-series score for the best run for each of the top 10 groups that participated in TREC 2004.

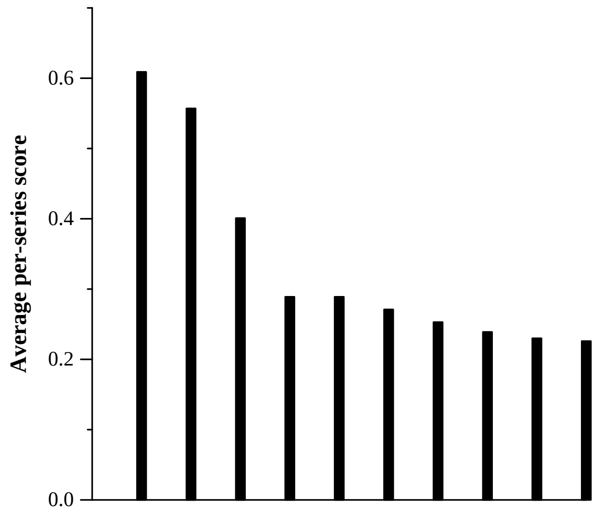


Figure 2: Average per-series scores for top ten QA track runs.

3 Analysis of Per-series Evaluation

The main purpose of evaluations such as TREC is to provide system builders with the information needed to improve their systems. An informative evaluation must be reliable (i.e., the results must be trustworthy) as well as capture salient aspects of the real user task. This section first examines the user task abstracted by the per-series evaluation, and then derives an empirical estimate of the reliability of the evaluation.

3.1 Modeling a User Task

The set of questions used to aggregate individual questions' scores determines the emphasis of a QA evaluation. In the TREC 2003 combined task there were no series but there were different question types, so question scores were first averaged by question type and then those averages were combined. This strategy emphasizes question-type analysis in that it is easy to compare different systems' abilities for the different question types. The QACIAD challenge contained only a single question type but introduced a series structure into the test set (Kato et al., 2004). In QACIAD, the scores were aggregated over the series and the series scores averaged. The QACIAD series were specifically constructed to be an abstraction of an information seeker's dialogue, and the aggregation of scores over series supports comparing different series types. For example, QACIAD results show browsing series to be more difficult than gathering series.

The TREC 2004 QA track contained both series structure and different question types, so individual question scores could be aggregated either by series or by question type. In general, the two methods of aggregation lead to different final scores. Aggregating by question type gives equal weight to

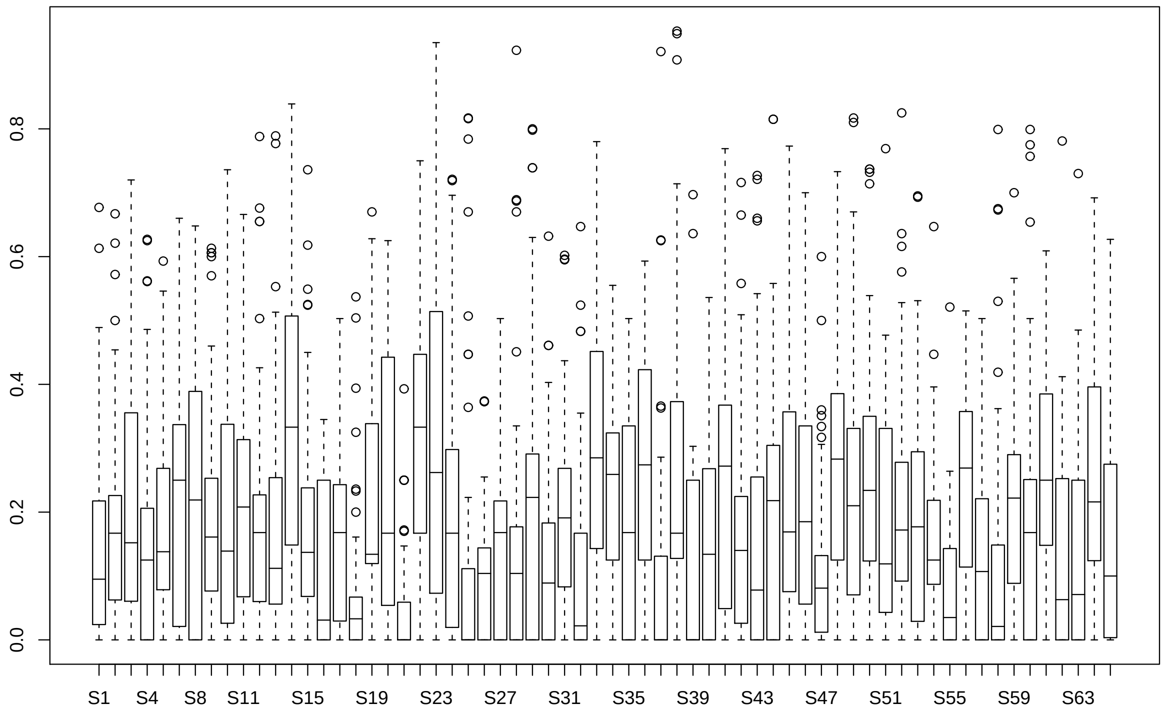


Figure 3: Box and whiskers plot of per-series scores across all TREC 2004 runs. The x-axis shows the series number and the y-axis the score.

each of the questions of the same type, while aggregating by series gives equal weight to each series. This is the same difference as between micro- and macro-averaging of document retrieval scores. For the set of runs submitted to TREC 2004, the absolute value of the final scores when aggregated by series were generally somewhat greater than the final scores when aggregated by question type, though it is possible for the question-type-aggregated score to be the greater of the two. The relative scores for different runs (i.e., whether one run was better than another) were usually, but not always, the same regardless of which aggregation method was used. The Kendall τ (Stuart, 1983) measure of correlation between the system rankings produced by sorting the runs by final score for each of the two aggregation methods was 0.971, where identical rankings would have a correlation of 1.0.

Despite the relatively minor differences in runs' final scores when aggregating by series or by question type, there is a strong reason to prefer the series aggregation. An individual series is small enough to be meaningful at the task level (it represents a single user's interaction) yet large enough for a series score to be meaningful. Figure 3 shows a box-and-whiskers plot of the per-series scores across all runs for each series. A box in the plot shows the extent of the middle half of the scores for that series, with the median score indicated by the line through the box. The dotted lines (the "whiskers") extend to a point that is 1.5 times the interquartile distance, or the most extreme score, whichever is less. Extreme scores that are greater than the 1.5 times the interquartile distance are plotted as circles. The plot shows that only a few series (S21, S25, S37, S39) have median scores of 0.0. This is in sharp contrast to the median scores of individual questions. For the TREC 2004 test set, 212 of the 230 factoid questions (92.2%) have a zero median, 39 of 55 list questions (70.9%) have a zero median, and 41 of 64 Other questions (64.1%) have a zero median.

Having a unit of evaluation that is at the appropriate level of granularity is necessary for meaningful results from the methodology used to assess the reliability of an evaluation. This methodology, described below, was originally created for document retrieval evaluations (Voorhees and Buckley, 2002) where the topic is the unit of evaluation. The distribution of scores across runs for an individual topic is much the same as the distribution of scores for the individual series as in figure 3. Score distributions that are heavily skewed toward zero make the evaluation look far more reliable than is likely to be the case since the reliability methodology computes a measure of the variability in scores.

3.2 Reliability

TREC uses comparative evaluations: one system is considered to be more effective than another if the evaluation score computed for the output of the first system is greater than the evaluation score computed for the output of the second system. Since all measurements have some (unknown) amount of error associated with them, there is always a chance that such a comparison can lead to the wrong result. An analysis of the reliability of an evaluation establishes bounds for how likely it is for a single comparison to be in error.

The reliability analysis uses the runs submitted to the track to empirically determine the relationship among the number of series in a test set, the observed difference in scores (δ) between two runs, and the likelihood that a single comparison of two runs leads to the correct conclusion. Once established, the relationship is used to derive the minimum difference in scores required for a certain level of confidence in the results given that there are 64 series in the test set.

The core of the procedure for establishing the relationship is comparing the effectiveness of a pair runs on two disjoint, equal-sized sets of series to see if the two sets disagree as to which of the runs is better. We define the error rate as the percentage of comparisons that have such a disagreement. Since the TREC 2004 track had 64 series, we can directly compute the error rate for test sizes up to 32 series. The smallest test set used is five series since fewer than five series in a test set is too noisy to be informative. By fitting curves to the values observed for test set sizes between 5 and 32, we can extrapolate the error rates to test sets up to 64 series.

When calculating the error rate, the difference between two runs' scores is categorized into a set of bins based on the size of the difference. The first bin contains runs with a difference of less than 0.01 (including no difference at all). The next bin contains runs whose difference is at least 0.01 but less than 0.02. The limits for the remaining bins increase by increments of 0.01.

Each test set size from 5 to 32 is treated as a separate experiment. Within an experiment, we randomly select two disjoint sets of series of the required size. We compute the average series score over both sets for all runs, then count the number of times we see a disagreement as to which run is better for all pairs of runs using the bins to segregate the counts by size of the difference in scores. The entire procedure is repeated 50 times (i.e., we perform 50 trials), with the counts of the number of disagreements kept as running totals over all trials. The ratio of the number of disagreements observed in a bin to the total number of cases that land in that bin is the error rate for the bin.

Figure 4 shows the error rate curves for five separate bins. In the figure the test set size is plotted on the x-axis and the error rate is plotted on the y-axis. The individual points in the graphs are the data points actually computed by the procedure above, while the lines are the best-fit exponential curve for the data points in the current bin and extrapolated to size 64. The top curve is for the bin with $0.01 \leq \delta < 0.02$ and the bottom curve for the bin with $0.05 \leq \delta < 0.06$; the intervening curves are for the intervening bins, in order with smaller δ's having larger error rates. An error rate no greater than 5%, requires a difference in scores of at least 0.05, which can be obtained with a test set of 47 series. Score differences of between 0.04 and 0.05 (the fourth curve) have an error rate slightly greater than 5% when there are 64 series in the test set.

Having established the minimum size of the difference in scores needed to be confident that two runs are actually different, it is also important to know whether differences of the required size actually occur in practice. If it is rare to observe a difference in scores as large as the minimum, then the evaluation will be reliable but insensitive. With 64 runs submitted to the TREC 2004 QA track, there are 1953 run pairs; 70% of the pairs have a difference in average per-series score that is at least 0.05. Many of the pairs in the remaining 30% are truly equivalent—for example, runs submitted by the same group that had very small differences in their processing. In figure 2, the difference in scores

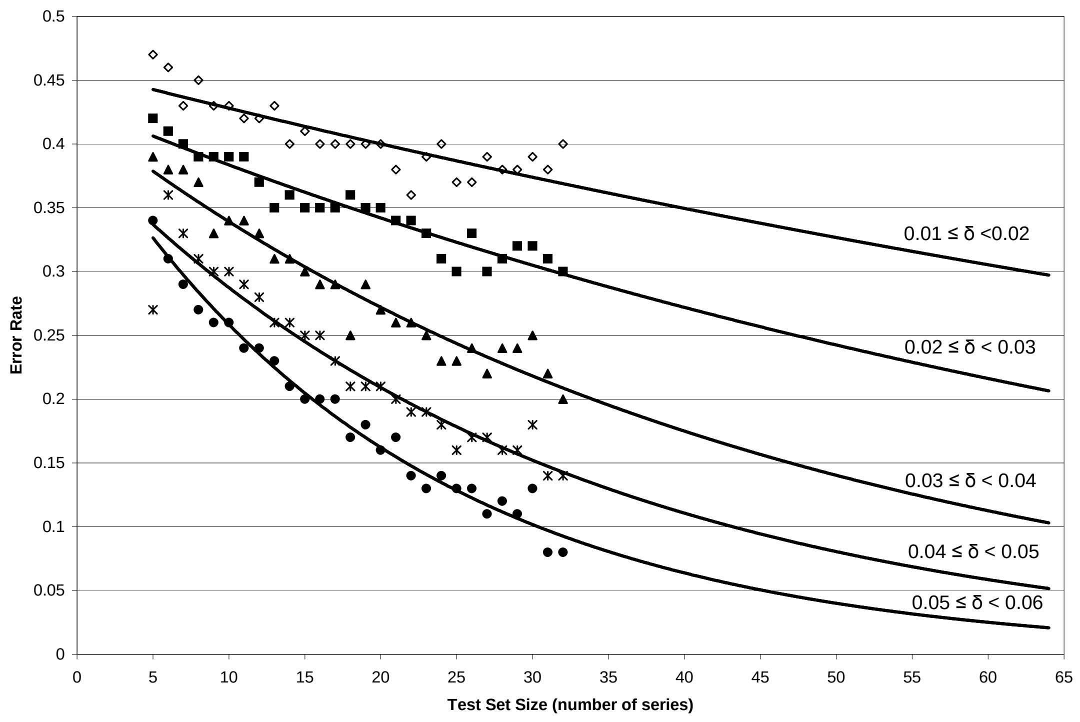


Figure 4: Extrapolated error rates for per-series scores for different test set sizes.

between each of the first three runs and its next closest run is greater than 0.05, while the next five runs are all within 0.05 of one another.

4 Conclusion

Question series have been introduced into recent question answering evaluations as a means of modeling dialogues between questioners and systems. The abstraction allows researchers to investigate methods for answering contextualized questions and for tracking (some forms of) the way objects are referred to in natural dialogues. The series have an important evaluation benefit as well. The individual series is at the correct level of granularity for aggregating scores for a meaningful evaluation. Unlike individual questions that have heavily skewed score distributions across runs, per-series score distributions resemble the distributions of per-topic scores in document retrieval evaluations. This allows the methodology developed for assessing the quality of a document retrieval evaluation to be meaningfully applied to the per-series evaluation. Such an analysis of the TREC 2004 QA track per-series evaluation shows the evaluation results to be reliable for differences in scores that are often observed in practice.

References

Tsuneaki Kato, Jun'ichi Fukumoto, Fumito Masui, and Noriko Kando. 2004. Handling information access dialogue through QA technologies—A novel challenge for open-domain question answering. In *Proceedings of the HLT-NAACL 2004 Workshop on Pragmatics of Question Answering*, pages 70–77, May.

Alan Stuart. 1983. Kendall's tau. In Samuel Kotz and Norman L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 4, pages 367–369. John Wiley & Sons.

Ellen M. Voorhees and Chris Buckley. 2002. The effect of topic set size on retrieval experiment error. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 316–323.

Ellen M. Voorhees. 2003. Evaluating answers to definition questions. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), Volume 2*, pages 109–111, May.

Ellen M. Voorhees. 2004. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, pages 54–68.

Combining Deep Linguistics Analysis and Surface Pattern Learning: A Hybrid Approach to Chinese Definitional Question Answering

Fuchun Peng, Ralph Weischedel, Ana Licuanan, Jinxi Xu
BBN Technologies
50 Moulton Street, Cambridge, MA, 02138
{fpeng, rweisched, alicuan, jxu}@bbn.com

Abstract

We explore a hybrid approach for Chinese definitional question answering by combining deep linguistic analysis with surface pattern learning. We answer four questions in this study: 1) How helpful are linguistic analysis and pattern learning? 2) What kind of questions can be answered by pattern matching? 3) How much annotation is required for a pattern-based system to achieve good performance? 4) What linguistic features are most useful? Extensive experiments are conducted on biographical questions and other definitional questions. Major findings include: 1) linguistic analysis and pattern learning are complementary; both are required to make a good definitional QA system; 2) pattern matching is very effective in answering biographical questions while less effective for other definitional questions; 3) only a small amount of annotation is required for a pattern learning system to achieve good performance on biographical questions; 4) the most useful linguistic features are copulas and appositives; relations also play an important role; only some propositions convey vital facts.

1 Introduction

Due to the ever increasing large amounts of online textual data, learning from textual data is becoming more and more important. Traditional document retrieval systems return a set of relevant documents and leave the users to locate the specific information they are interested in. Question answering, which combines traditional document retrieval and information extraction, solves this problem directly by returning users the specific answers. Research in textual question answering has made substantial advances in the past few years (Voorhees, 2004).

Most question answering research has been focusing on factoid questions where the goal is to return a list of facts about a concept. Definitional questions, however, remain largely unexplored. Definitional questions differ from factoid questions in that the goal is to return the relevant "answer nuggets" of information about a query. Identifying such answer nuggets requires more advanced language processing techniques. Definitional QA systems are not only interesting as a research challenge. They also have the potential to be a valuable complement to static knowledge sources like encyclopedias. This is because they create definitions dynamically, and thus answer definitional questions about terms which are new or emerging (Blair-Goldensoha et al., 2004).

One success in factoid question answering is pattern based systems, either manually constructed (Soubbotin and Soubbotin, 2002) or machine learned (Cui et al., 2004). However, it is unknown whether such pure pattern based systems work well on definitional questions where answers are more diverse.

Deep linguistic analysis has been found useful in factoid question answering (Moldovan et al., 2002) and has been used for definitional questions (Xu et al., 2004; Harabagiu et al., 2003). Linguistic analysis is useful because full parsing captures long distance dependencies between the answers and the query terms, and provides more information for inference. However, merely linguistic analysis may not be enough. First, current state of the art linguistic analysis such as parsing, co-reference, and relation extraction is still far below human performance. Errors made in this stage will propagate and lower system accuracy. Second, answers to some types of definitional questions may have strong local dependencies that can be better captured by surface patterns. Thus we believe that combining linguistic analysis and pattern learning would be complementary and be beneficial to the whole system.

Work in combining linguistic analysis with patterns include Weischedel et al. (2004) and Jijkoun et al. (2004) where manually constructed patterns are used to augment linguistic features. However, manual pattern construction critically depends on the domain knowledge of the pattern designer and often has low coverage (Jijkoun et al., 2004). Automatic pattern derivation is more appealing (Ravichandran and Hovy, 2002).

In this work, we explore a hybrid approach to combining deep linguistic analysis with automatic pattern learning. We are interested in answering the following four questions for Chinese definitional question answering:

- How helpful are linguistic analysis and pattern learning in definitional question answering?
- If pattern learning is useful, what kind of question can pattern matching answer?
- How much human annotation is required for a pattern based system to achieve reasonable performance?
- If linguistic analysis is helpful, what linguistic features are most useful?

To our knowledge, this is the first formal study of these questions in Chinese definitional QA. To answer these questions, we perform extensive experiments on Chinese TDT4 data (Linguistic Data Consortium, 2002-2003). We separate definitional questions into biographical (Who-is) questions and other definitional (What-is) questions. We annotate some question-answer snippets for pattern learning and we perform deep linguistic analysis including parsing, tagging, name entity recognition, co-reference, and relation detection.

2 A Hybrid Approach to Definitional Question Answering

The architecture of our QA system is shown in Figure 1. Given a question, we first use simple rules to classify it as a “Who-is” or “What-is” question and detect key words. Then we use a HMM-based IR system (Miller et al., 1999) for document retrieval by treating the question keywords as a query. To speed up processing, we only use the top 1000 relevant documents. We then select relevant sentences among the returned relevant documents. A sentence is considered relevant if it contains the query keyword or contains a word that is co-referent to the query term. Coreference is determined using an information extraction engine, SERIF (Ramshaw et al., 2001). We then conduct deep linguistic analysis and pattern matching to extract candidate answers. We rank all candidate answers by predetermined feature ordering. At the same time, we perform redundancy detection based on n-gram overlap.

2.1 Deep Linguistic Analysis

We use SERIF (Ramshaw et al., 2001), a linguistic analysis engine, to perform full parsing, name entity detection, relation detection, and co-reference resolution. We extract the following linguistic features:

1. Copula: a copula is a linking verb such as “*is*” or “*become*”. An example of a copula feature is “*Bill Gates is the CEO of Microsoft*”. In this case, “*CEO of Microsoft*” will be extracted as an answer to “*Who is Bill Gates?*”. To extract copulas, SERIF traverses the parse trees of the sentences and extracts copulas based on rules. In Chinese, the rule for identifying a copula is the POS tag “*VC*”, standing for “*Verb Copula*”. The only copula verb in Chinese is “是”.

2. Apposition: appositions are a pair of noun phrases in which one modifies the other. For example, In “*Tony Blair, the British Prime Minister, ...*”, the phrase “*the British Prime Minister*” is in apposition to “*Blair*”. Extraction of appositive features is similar to that of copula. SERIF traverses the parse tree and identifies appositives based on rules. A detailed description of the algorithm is documented

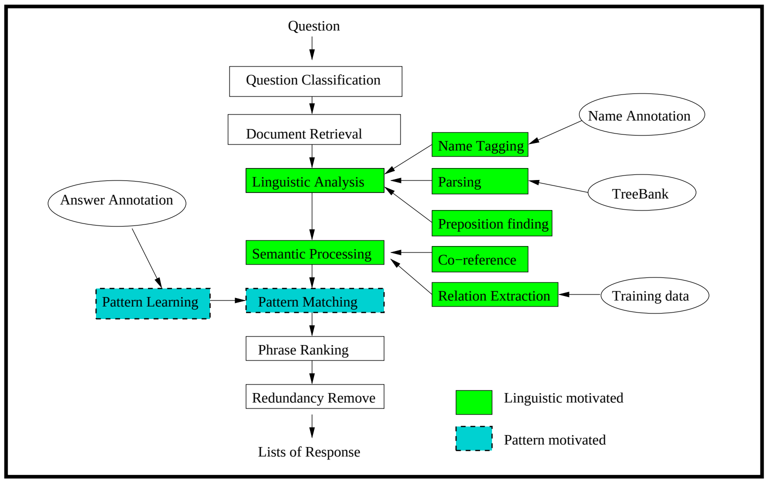


Figure 1: Question answering system structure

in (Ramshaw et al., 2001).

3. Proposition: propositions represent predicate-argument structures and take the form: *predicate(*$role_1$*:* arg_1*, ...,* $role_n$*:* arg_n*)*. The most common roles include logical subject, logical object, and object of a prepositional phrase that modifies the predicate. For example, "Smith went to Spain" is represented as a proposition, *went(logical subject: Smith, PP-to: Spain)*.

4. Relations: The SERIF linguistic analysis engine also extracts relations between two objects. SERIF can extract 24 binary relations defined in the ACE guidelines (Linguistic Data Consortium, 2002), such as spouse-of, staff-of, parent-of, management-of and so forth. Based on question types, we use different relations, as listed in Table 1.

Relations used for Who-Is questions
ROLE/MANAGEMENT, ROLE/GENERAL-STAFF, ROLE/CITIZEN-OF, ROLE/FOUNDER, ROLE/OWNER, AT/RESIDENCE, SOC/SPOUSE, SOC/PARENT, ROLE/MEMBER, SOC/OTHER-PROFESSIONAL
Relation used for What-Is questions
AT/BASED-IN, AT/LOCATED, PART/PART-OF

Table 1: Relations used in our system

Many relevant sentences do not contain the query key words. Instead, they contain words that are co-referent to the query. For example, in "*Yesterday UN Secretary General* **Anan** *Requested Every Side...,* **He** *said ...* ". The pronoun "He" in the second sentence refers to "Anan" in the first sentence. To select such sentences, we conduct co-reference resolution using SERIF.

In addition, SERIF also provides name tagging, identifying 29 types of entity names or descriptions, such as locations, persons, organizations, and diseases.

We also select complete sentences mentioning the term being defined as backup answers if no other features are identified.

The component performance of our linguistic analysis is shown in Table 2.

	Pre.	Recall	F
Parsing	0.813	0.828	0.820
Co-reference	0.920	0.897	0.908
Name-entity detection	0.765	0.753	0.759

Table 2: Linguistic analysis component performance for Chinese

2.2 Surface Pattern Learning

We use two kinds of patterns: manually constructed patterns and automatically derived patterns. A manual pattern is a commonly used linguistic expression that specifies aliases, super/subclass and membership relations of a term (Xu et al., 2004). For example, the expression "*tsunamis, also known as tidal waves*" gives an alternative term for tsunamis. We use 23 manual patterns for *Who-is* questions and 14 manual patterns for *What-is* questions.

We also classify some special propositions as manual patterns since they are specified by computational linguists. After a proposition is extracted, it is matched against a list of predefined predicates. If it is on the list, it is considered special and will be ranked higher. In total, we designed 22 special propositions for *Who-is* questions, such as 成为(become), 当选为(elected as), and 辞去(resign), 14 for *What-is* questions, such as 位于(located at), 创建于(created at), and 又称为(also known as).

However, it is hard to manually construct such patterns since it largely depends on the knowledge of the pattern designer. Thus, we prefer patterns that can be automatically derived from training data. Some annotators labeled question-answer snippets. Given a query question, the annotators were asked to highlight the strings that can answer the question. Though such a process still requires annotators to have knowledge of what can be answers, it does not require a computational linguist. Our pattern learning procedure is illustrated in Figure 2.

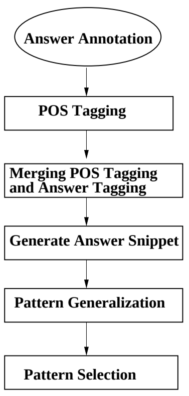


Figure 2: Surface Pattern Learning

Here we give an example to illustrate how pattern learning works. The first step is annotation. An example of Chinese answer annotation with English translation is shown in Figure 3. Question words are assigned the tag *QTERM*, answer words are tagged *ANSWER*, and all other words are assigned *BKGD*, standing for background words (not shown in the example to make the annotation more readable).

To obtain patterns, we conduct full parsing to obtain the full parse tree for a sentence. In our current

Chinese annotation: 到朝鲜进“行破冰之旅”的(美国国务卿 ANSWER)(奥尔布赖特 QTERM), 昨天同朝鲜领袖今正日进行历史性会谈

English translation: (U.S. Secretary of the State ANWER) (Albright QTERM), who visited North Korea for the “ice-breaking trip”, had a historical meeting with the leader of North Korea, Kim Jong Il.

Figure 3: Answer annotation example

patterns, we only use POS tagging information, but other higher level information could also be used. The segmented and POS tagged sentence is shown in Figure 4. Each word is assigned a POS tag as defined by the Penn Chinese Treebank guidelines.

(到 P)(朝 鲜 NR)(进 行 VV)(“ PU)(破 VV)(冰 NN)(之 旅 NN)(” PU)(的 DEC)(美 国 NR)(国 务 卿 NR)(奥 尔 NR)(布 赖 特 NR)(, PU) (昨 天 NT)(同 DT)(朝鲜 NR)(领袖 NR)(今正日 NN)(举行 VV)(历史性 JJ)(会谈 NN).

Figure 4: POS tagging

Next we combine the POS tags and the answer tags by appending these two tags to create a new tag, as shown in Figure 5.

(到 P/BKGD)(朝 鲜 NR/BKGD)(进 行 VV/BKGD)(“ PU/BKGD)(破 VV/BKGD)(冰 NN/BKGD)(之 旅 NN/BKGD)(” PU/BKGD)(的 DEC/BKGD)(美 国 NR/ANSWER)(国 务 卿 NR/ANSWER)(奥 尔 NR/QTERM)(布 赖 特 NR/QTERM)(, PU/BKGD) (昨 天 NT/BKGD)(同 DT/BKGD)(朝 鲜 NR/BKGD)(领 袖 NR/BKGD)(金正日 NN/BKGD)(进行 VV/BKGD)(历史性 JJ/BKGD)(会谈 NN/BKGD)

Figure 5: Combined POS and Answer tagging

We can then obtain an answer snippet from this training sample. Here we obtain the snippet *(美国国务卿 NR/ANSWER)(TERM)*.

We generalize a pattern using three heuristics (this particular example does not generalize). First, we replace all Chinese sequences longer than 3 characters with their POS tags, under the theory that long sequences are too specific. Second, we also replace NT (time noun, such as 昨天), DT (determiner, such as 这, 那), cardinals (CD, such as 一, 二, 三) and M (measurement word such as 年月日) with their POS tags. Third, we ignore adjectives.

After obtaining all patterns, we run them on the training data to calculate their precision and recall. We select patterns whose precision is above 0.6 and which fire at least 5 times in training data (parameters are determined with a held out dataset).

3 Experiments

3.1 Data Sets

We produced a list of questions and asked annotators to identify answer snippets from TDT4 data. To produce as many training answer snippets as possible, annotators were asked to label answers exhaustively; that is, the same answer can be labeled multiple times in different places. However, we remove duplicate answers for test questions since we are only interested in unique answers in evaluation.

We separate questions into two types, biographical (Who-is) questions, and other definitional questions (What-is). For "Who-is" questions, we used 204 questions for pattern learning, 10 for parameter tuning and another 42 questions for testing. For "What-is" questions, we used 44 for training and another 44 for testing.

3.2 Evaluation

The TREC question answering evaluation is based on human judgments (Voorhees, 2004). However, such a manual procedure is costly and time consuming. Recently, researchers have started automatic question answering evaluation (Xu et al., 2004; Lin and Demner-Fushman, 2005; Soricut and Brill, 2004). We use Rouge, an automatic evaluation metric that was originally used for summarization evaluation (Lin and Hovy, 2003) and was recently found useful for evaluating definitional question answering (Xu et al., 2004). Rouge is based on n-gram co-occurrence. An n-gram is a sequence of n consecutive Chinese characters.

Given a reference answer R and a system answer S, the Rouge score is defined as follows:

$$Rouge(R,S,N) = \sqrt[N]{\prod_{n=1}^{N} \frac{count_{match}(R,S,n)}{count(R,n)}}$$

where N is the maximum length of n-grams, $count_{match}(R,S,n)$ is the number of common n-grams of R and S, and $count(R,n)$ is the number of n-grams in R. If N is too small, stop words and bi-grams of such words will dominate the score; If N is too large, there will be many questions without answers. We select N to be 3, 4, 5 and 6.

To make scores of different systems comparable, we truncate system output for the same question by the same cutoff length. We score answers truncated at length L times that of the reference answers, where L is set to be 1, 2, and 3. The rationale is that people would like to read at least the same length of the reference answer. On the other hand, since the state of the art system answer is still far from human performance, it is reasonable to produce answers somewhat longer than the references (Xu et al., 2004).

In summary, we run experiments with parameters $N = 3,4,5,6$ and $L = 1,2,3$, and take the average over all of the 12 runs.

3.3 Overall Results

We set the pure linguistic analysis based system as the baseline and compare it to other configurations. Table 3 and Table 4 show the results on "*Who-is*" and "*What-is*" questions respectively. The baseline (Run 1) is the result of using pure linguistic features; Run 2 is the result of adding manual patterns to the baseline system; Run 3 is the result of using learned patterns only. Run 4 is the result of adding learned patterns to the baseline system. Run 5 is the result of adding both manual patterns and learned patterns to the system.

The first question we want to answer is how helpful the linguistic analysis and pattern learning are for definitional QA. Comparing Run 1 and 3, we can see that both pure linguistic analysis and pure pattern based systems achieve comparable performance; Combining them together improves performance (Run 4) for "who is" questions, but only slightly for "what is" questions. This indicates that linguistic analysis and pattern learning are complementary to each other, and both are helpful for biographical QA.

The second question we want to answer is what kind of questions can be answered with pattern matching. From these two tables, we can see that patterns are very effective in "*Who-is*" questions while less effective in "*What-is*" questions. Learned patterns improve the baseline from 0.3399 to 0.3860; manual patterns improve the baseline to 0.3657; combining both manual and learned patterns improve it to 0.4026, an improvement of **18.4%** compared to the baseline. However, the effect of patterns on "*What-is*" is smaller, with an improvement of only **3.5%**. However, the baseline performance on "*What-is*" is also much worse than that of "*Who-is*" questions. We will analyze the reasons in Section 4.3. This indicates that answering general definitional questions is much more challenging than answering biographical questions and deserves more research.

Run	Run description	Rouge
(1)	Baseline	0.3399
(2)	(1)+ manual patterns	0.3657
(3)	Learned patterns	0.3549
(4)	(1)+ learned patterns	0.3860
(5)	(2)+ learned patterns	0.4026

Table 3: Results on *Who-is* (Biographical) Questions

Run	Run description	Rouge
(1)	Baseline	0.2126
(2)	(1)+ manual patterns	0.2153
(3)	Learned patterns	0.2117
(4)	(1)+ learned patterns	0.2167
(5)	(2)+ learned patterns	0.2201

Table 4: Results on "*What-is*" (Other Definitional) Questions

4 Analysis

4.1 How much annotation is needed

The third question is how much annotation is needed for a pattern based system to achieve good performance. We run experiments with portions of training data on biographical questions, which produce different number of patterns. Table 5 shows the details of the number of training snippets used and the number of patterns produced and selected. The performance of different system is illustrated in Figure 6. With only 10% of the training data (549 snippets, about two person days of annotation), learned patterns achieve good performance of 0.3285, considering the performance of 0.3399 of a well tuned system with deep linguistic features. Performance saturates with 2742 training snippets (50% training, 10 person days annotation) at a Rouge score of 0.3590, comparable to the performance of a well tuned system with full linguistic features and manual patterns (Run 2 in Table 3). There could even be a slight, insignificant performance decrease with more training data because our sampling is sequential instead of random. Some portions of training data might be more useful than others.

	Training snippets	Patterns learned	Patterns selected
10% train	549	56	33
30% train	1645	144	88
50% train	2742	211	135
70% train	3839	281	183
90% train	4935	343	222
100% train	5483	381	266

Table 5: Number of patterns with different size of training data

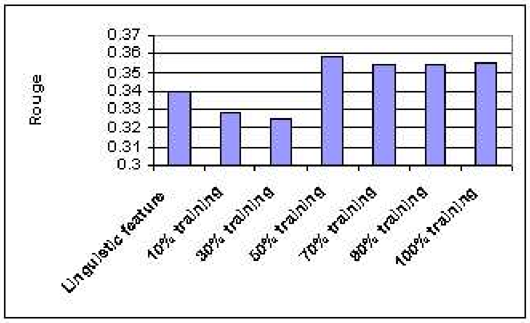


Figure 6: How much annotation is required (measured on biographical questions)

4.2 Contributions of different features

The fourth question we want to answer is: what features are most useful in definitional question answering? To evaluate the contribution of each individual feature, we turn off all other features and test the system on a held out data (10 questions). We calculate the coverage of each feature, measured by Rouge. We also calculate the precision of each feature with the following formula, which is very similar to Rouge except that the denominator here is based on system output $count(S, n)$ instead of reference $count(R, n)$. The notations are the same as those in Rouge.

$$Precision(R,S,N) = \sqrt[N]{\prod_{n=1}^{N} \frac{count_{match}(R,S,n)}{count(S,n)}}$$

Figure 7 is the precision-recall scatter plot of the features measured on "who is" questions. Interestingly, the learned patterns have the highest coverage and precision. The copula feature has the second highest precision; however, it has the lowest coverage. This is because there are not many copulas in the dataset. Appositive and manual pattern features have the same level of contribution. Surprisingly, the relation feature has a high coverage. This suggests that relations could be more useful if relation detection were more accurate; general propositions are not more useful than whole sentences since almost every sentence has a proposition, and since the high value propositions are identified by the lexical head of the proposition and grouped with the manual patterns.

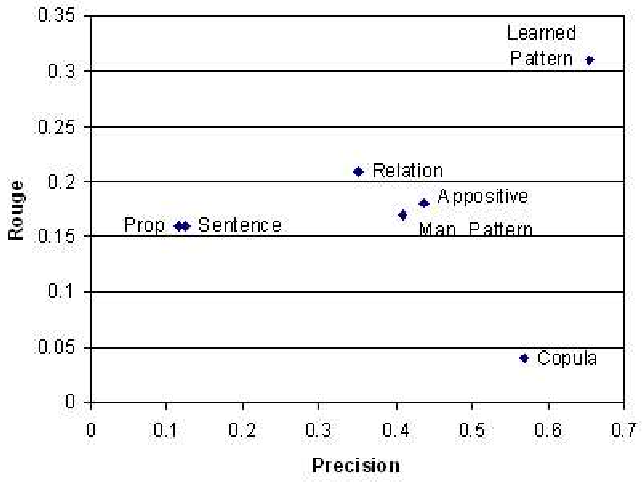


Figure 7: Feature precision recall scatter plot (measured on the biographical questions)

4.3 Who-is versus What-is questions

We have seen that "*What-is*" questions are more challenging than "*Who-is*" questions. We compare the precision and coverage of each feature for "*Who-is*" and "*What-is*" in Table 6 and Table 7. We see that although the precisions of the features are higher for "*What-is*", their coverage is too low. The most useful features for "*What-is*" questions are propositions and raw sentences, which are the worst two features for "*Who-is*". Basically, this means that most of the answers for "*What-is*" are from whole sentences. Neither linguistic analysis nor pattern matching works as efficiently as in biographical questions.

feature	who-is	what-is
copula	0.567	0.797
appositive	0.3460	0.3657
proposition	0.1162	0.1837
relation	0.3509	0.4422
sentence	0.1074	0.1556
learned patterns	0.6542	0.6858

Table 6: Feature Precision Comparison

feature	who-is	what-is
copula	0.055	0.049
appositive	0.2028	0.0026
proposition	0.2101	0.1683
relation	0.2722	0.043
sentence	0.1619	0.1717
learned patterns	0.3517	0.0860

Table 7: Feature Coverage Comparison

To identify the challenges of "*What-is*" questions, we conducted an error analysis. The answers for "*What-is*" are much more diverse and are hard to capture. For example, the reference answers for the question of "什么是国际空间站？/ What is the international space station?" include the weight of the space station, the distance from the space station to the earth, the inner structure of the space station, and the cost of its construction. Such attributes are hard to capture with patterns, and they do not contain any of the useful linguistic features we currently have (copula, appositive, proposition, relation). Identifying more useful features for such answers remains for future work.

5 Related Work

Ravichandran and Hovy (2002) presents a method that learns patterns from online data using some seed questions and answer anchors. The advantage is that it does not require human annotation. However, it only works for certain types of questions that have fixed anchors, such as “where was X born”. For general definitional questions, we do not know what the anchors should be. Thus we prefer using small amounts of human annotation to derive patterns. Cui et al. (2004) uses a similar approach for unsupervised pattern learning and generalization to soft pattern matching. However, the method is actually used for sentence selection rather than answer snippet selection. Combining information extraction with surface patterns has also seen some success. Jikoun et al. (2004) shows that information extraction can help improve the recall of a pattern based system. Xu et al. (2004) also shows that manually constructed patterns are very important in answering English definitional questions. Hildebrandt et al. (2004) uses manual surface patterns for target extraction to augment database and dictionary lookup. Blair-Goldensohn et al. (2004) apply supervised learning for definitional predicates and then apply summarization methods for question answering.

6 Conclusions and Future Work

We have explored a hybrid approach for definitional question answering by combining deep linguistic analysis and surface pattern learning. For the first time, we have answered four questions regarding Chinese definitional QA: deep linguistic analysis and automatic pattern learning are complementary and may be combined; patterns are powerful in answering biographical questions; only a small amount of annotation (2 days) is required to obtain good performance in a biographical QA system; copulas and appositions are the most useful linguistic features; relation extraction also helps.

Answering “*What-is*” questions is more challenging than answering “*Who-is*” questions. To improve the performance on “*What-is*” questions, we could divide “*What-is*” questions into finer classes such as organization, location, disease, and general substance, and process them specifically.

Our current pattern matching is based on simple POS tagging which captures only limited syntactic information. We generalize words to their corresponding POS tags. Another possible improvement is to generalize using automatically derived word clusters, which provide semantic information.

Acknowledgements This material is based upon work supported by the Advanced Research and Development Activity (ARDA) under Contract No. NBCHC040039. We are grateful to Linnea Micciulla for proof reading and three anonymous reviewers for suggestions on improving the paper.

References

S. Blair-Goldensoha, K. McKeown, and A. Hazen Schlaikjer. 2004. Answering Definitional Questions: A Hybrid Approach. *New Directions In Question Answering.*, pages 47–58.

H. Cui, M. Kan, and T. Chua. 2004. Unsupervised Learning of Soft Patterns for Definitional Question Answering. In *WWW 2004*, pages 90–99.

S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams, and J. Bensley. 2003. Answer Mining by Combining Extraction Techniques with Abductive Reasoning. In *TREC2003 Proceedings*.

W. Hildebrandt, B. Katz, and J. Lin. 2004. Answering Definition Questions with Multiple Knowledge Sources. In *HLT-NAACL 2004*, pages 49–56.

V. Jijkoun, M. Rijke, and J. Mur. 2004. Information Extraction for Question Answering: Improving Recall Through Syntactic Patterns. In *COLING 2004*.

J. Lin and D. Demner-Fushman. 2005. Automatically Evaluating Answers to Definition Questions. In *ACL2005*. to appear.

C. Lin and E. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *HLT-NAACL 2003*.

D. Miller, T. Leek, and R. Schwartz. 1999. A Hidden Markov Model Information Retrieval System. In *SIGIR 1999*, pages 214 – 221.

D. Moldovan, M. Pasca, S. Harabagiu, and M. Surdeanu. 2002. Performance Issues and Error Analysis in an Open-Domain Question Answering System. In *ACL2002*.

L. Ramshaw, E. Boshee, S. Bautus, S. Miller, R. Stone, R. Weischedel, and A. Zamanian. 2001. Experiments in Multi-Model Automatic Content Extraction. In *HLT2001*.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a Question Answering System. In *ACL2002*, pages 41–47.

R. Soricut and E. Brill. 2004. A Unified Framework For Automatic Evaluation Using N-Gram Co-occurrence Statistics. In *ACL 2004*, pages 613–620.

M. Soubbotin and S. Soubbotin. 2002. Use of Patterns for Detection of Likely Answer Strings: A Systematic Approach. In *TREC2002 Proceedings*.

E. Voorhees. 2004. Overview of the TREC 2003 Question Answering Track. In *TREC Proceedings*.

R. Weischedel, J. Xu, and A. Licuanan. 2004. A Hybrid Approach to Answering Biographical Questions. *New Directions In Question Answering.*, pages 59–70.

J. Xu, R. Weischedel, and A. Licuanan. 2004. Evaluation of an Extraction-based Approach to Answering Definitional Questions. In *SIGIR 2004*, pages 418–424.

Enhanced Answer Type Inference from Questions using Sequential Models

Vijay Krishnan and **Sujatha Das** and **Soumen Chakrabarti**[*]
Computer Science and Engineering Department, IIT Bombay, India

Abstract

Question classification is an important step in factual question answering (QA) and other dialog systems. Several attempts have been made to apply statistical machine learning approaches, including Support Vector Machines (SVMs) with sophisticated features and kernels. Curiously, the payoff beyond a simple bag-of-words representation has been small. We show that most questions reveal their class through a short contiguous token subsequence, which we call its *informer span*. Perfect knowledge of informer spans can enhance accuracy from 79.4% to 88% using linear SVMs on standard benchmarks. In contrast, standard heuristics based on shallow pattern-matching give only a 3% improvement, showing that the notion of an informer is non-trivial. Using a novel multi-resolution encoding of the question's parse tree, we induce a Conditional Random Field (CRF) to identify informer spans with about 85% accuracy. Then we build a meta-classifier using a linear SVM on the CRF output, enhancing accuracy to 86.2%, which is better than all published numbers.

1 Introduction

An important step in factual question answering (QA) and other dialog systems is to classify the question (e.g., Who painted Olympia?) to the anticipated type of the answer (e.g., person). This step is called "question classification" or "answer type identification".

The answer type is picked from a hand-built taxonomy having dozens to hundreds of answer types (Harabagiu et al., 2000; Hovy et al., 2001; Kwok et al., 2001; Zheng, 2002; Dumais et al., 2002). QA systems can use the answer type to short-list answer tokens from passages retrieved by an information retrieval (IR) subsystem, or use the type together with other question words to inject IR queries.

Early successful QA systems used manually-constructed sets of rules to map a question to a type, exploiting clues such as the wh-word (who, where, when, how many) and the head of noun phrases associated with the main verb (what <u>is</u> the <u>tallest</u> *<u>mountain</u>* in . . .).

With the increasing popularity of statistical NLP, Li and Roth (2002), Hacioglu and Ward (2003) and Zhang and Lee (2003) used supervised learning for question classification on a data set from UIUC that is now standard[1]. It has 6 coarse and 50 fine answer types in a two-level taxonomy, together with 5500 training and 500 test questions. Webclopedia (Hovy et al., 2001) has also published its taxonomy with over 140 types.

The promise of a machine learning approach is that the QA system builder can now focus on designing features and providing labeled data, rather than coding and maintaining complex heuristic rule-bases. The data sets and learning systems quoted above have made question classification a well-defined and non-trivial subtask of QA for which algorithms can be evaluated precisely, isolating more complex factors at work in a complete QA system.

Prior work: Compared to human performance, the accuracy of question classifiers is not high. In all studies, surprisingly slim gains have resulted from sophisticated design of features and kernels.

Li and Roth (2002) used a Sparse Network of Winnows (SNoW) (Khardon et al., 1999). Their features included tokens, parts of speech (POS), chunks (non-overlapping phrases) and named entity (NE) tags. They achieved 78.8% accuracy for 50 classes, which improved to 84.2% on using an (unpublished, to our knowledge) hand-built dictionary of "semantically related words".

[*] `soumen@cse.iitb.ac.in`

[1] `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 315–322, Vancouver, October 2005. ©2005 Association for Computational Linguistics

Hacioglu and Ward (2003) used linear support vector machines (SVMs) with question word 2-grams and error-correcting output codes (ECOC)—but no NE tagger or related word dictionary—to get 80.2–82% accuracy.

Zhang and Lee (2003) used linear SVMs with all possible question word q-grams, and obtained 79.2% accuracy. They went on to design an ingenious kernel on question parse trees, which yielded visible gains for the 6 coarse labels, but only "slight" gains for the 50 fine classes, because "the syntactic tree does not normally contain the information required to distinguish between the various fine categories within a coarse category".

Algorithm	6-class	50-class
Li and Roth, SNoW	(1)	78.8(2)
Hacioglu et al., SVM+ECOC	–	80.2–82
Zhang & Lee, LinearSVMq	87.4	79.2
Zhang & Lee, TreeSVM	90	–
SVM, "perfect" informer	**94.2**	**88**
SVM, CRF-informer	**93.4**	**86.2**

Table 1: Summary of % accuracy for UIUC data. (1) SNoW accuracy without the related word dictionary was not reported. With the related-word dictionary, it achieved 91%. (2) SNoW with a related-word dictionary achieved 84.2% but the other algorithms did not use it. Our results are summarized in the last two rows, see text for details.

Our contributions: We introduce the notion of the **answer type informer span** of the question (in §2): a short (typically 1–3 word) subsequence of question tokens that are adequate clues for question classification; e.g.: How much does an adult elephant *weigh*?

We show (in §3.2) that a simple linear SVM using features derived from human-annotated informer spans beats all known learning approaches. This confirms our suspicion that the earlier approaches suffered from a feature localization problem.

Of course, informers are useful only if we can find ways to automatically identify informer spans. Surprisingly, syntactic pattern-matching and heuristics widely used in QA systems are not very good at capturing informer spans (§3.3). Therefore, the notion of an informer is non-trivial.

Using a parse of the question sentence, we derive a novel set of multi-resolution features suitable for training a conditional random field (CRF) (Lafferty et al., 2001; Sha and Pereira, 2003). Our feature design paradigm may be of independent interest (§4). Our informer tagger is about 85–87% accurate.

We use a meta-learning framework (Chan and Stolfo, 1993) in which a linear SVM predicts the answer type based on features derived from the original question as well as the output of the CRF. This meta-classifier beats all published numbers on standard question classification benchmarks (§4.4). Table 1 (last two rows) summarizes our main results.

2 Informer overview

Our key insight is that a human can classify a question based on very few tokens gleaned from skeletal syntactic information. This is certainly true of the most trivial classes (*Who* wrote Hamlet? or *How many* dogs pull a sled at Iditarod?) but is also true of more subtle clues (How much does a rhino *weigh*?).

In fact, informal experiments revealed the surprising property that *only one* contiguous span of tokens is adequate for a human to classify a question. E.g., in the above question, a human does not even need the *how much* clue once the word *weigh* is available. In fact, "How much does a rhino *cost*?" has an identical syntax but a completely different answer type, not revealed by *how much* alone. The only exceptions to the single-span hypothesis are multi-function questions like "What is the *name* and *age* of . . .", which should be assigned to multiple answer types. In this paper we consider questions where one type suffices.

Consider another question with multiple clues: *Who* is the *CEO* of IBM? In isolation, the clue *who* merely tells us that the answer might be a person or country or organization, while *CEO* is perfectly precise, rendering *who* unnecessary. All of the above applies *a forteriori* to *what* and *which* clues, which are essentially uninformative on their own, as in "What is the *distance* between Pisa and Rome?"

Conventional QA systems use mild analysis on the wh-clues, and need much more sophistication on the rest of the question (e.g. inferring *author* from *wrote*, and even verb subcategorization). We submit that a single, minimal, suitably-chosen contiguous span of question token/s, defined as the **informer span** of the question, is adequate for question classification.

The informer span is very sensitive to the structure of clauses, phrases and possessives in the question, as is clear from these examples (informers italicized): "What is Bill Clinton's wife's *profession*", and "What *country*'s president was shot at Ford's Theater". The choice of informer spans also depends on the target classification system. Initially we wished to handle definition questions separately, and marked no informer tokens in "What is digitalis". However, *what is* is an excellent informer for the UIUC class `DESC:def` (description, definition).

3 The meta-learning approach

We propose a meta-learning approach (§3.1) in which the SVM can use features from the original question as well as its informer span. We show (§3.2) that human-annotated informer spans lead to large improvements in accuracy. However, we show (§3.3) that simple heuristic extraction rules commonly used in QA systems (e.g. head of noun phrase following wh-word) cannot provide informers that are nearly as useful. This naturally leads us to designing an informer tagger in §4.

Figure 1 shows our meta-learning (Chan and Stolfo, 1993) framework. The combiner is a linear multi-class one-vs-one SVM[2], as in the Zhang and Lee (2003) baseline. We did not use ECOC (Hacioglu and Ward, 2003) because the reported gain is less than 1%.

The word feature extractor selects unigrams and q-grams from the question. In our experience, $q = 1$ or $q = 2$ were best; if unspecified, all possible $q$grams were used. Through tuning, we also found that the SVM "C" parameter (used to trade between training data fit and model complexity) must be set to 300 to achieve their published baseline numbers.

3.1 Adding informer features

We propose two very simple ways to derive features from informers for use with SVMs. Initially, assume that perfect informers are known for all questions;

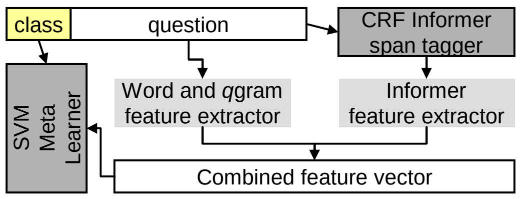


Figure 1: The meta-learning approach.

later (§4) we study how to predict informers.

Informer q-grams: This comprises of all word q-grams within the informer span, for all possible q. E.g., such features enable effective exploitation of informers like *length* or *height* to classify to the `NUMBER:distance` class in the UIUC data.

Informer q-gram hypernyms: For each word or compound within the informer span that is a WordNet noun, we add all hypernyms of all senses. The intuition is that the informer (e.g. *author*, *cricketer*, *CEO*) is often narrower than a broad question class (`HUMAN:individual`). Following hypernym links up to *person* via WordNet produces a more reliably correlated feature.

Given informers, other question words might seem useless to the classifier. However, retaining regular features from other question words is an excellent idea for the following reasons.

First, we kept word sense disambiguation (WSD) outside the scope of this work because WSD entails computation costs, and is unlikely to be reliable on short single-sentence questions. Questions like *How long* ... or *Which bank* ... can thus become ambiguous and corrupt the informer hypernym features. Additional question words can often help nail the correct class despite the feature corruption.

Second, while our CRF-based approach to informer span tagging is better than obvious alternatives, it still has a 15% error rate. For the questions where the CRF prediction is wrong, features from non-informer words give the SVM an opportunity to still pick the correct question class.

Word features: Based on the above discussion, one boolean SVM feature is created for every word q-gram over all question tokens. In experiments, we found bigrams ($q = 2$) to be most effective, closely followed by unigrams ($q = 1$). As with informers, we can also use hypernyms of regular words as SVM

[2]`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

features (marked “Question bigrams + hypernyms” in Table 2).

3.2 Benefits from “perfect” informers

We first wished to test the hypothesis that identifying informer spans to an SVM learner can improve classification accuracy. Over and above the class labels, we had two volunteers tag the 6000 UIUC questions with informer spans (which we call “perfect”—agreement was near-perfect).

Features	Coarse	Fine
Question trigrams	91.2	77.6
All question $q$grams	87.2	71.8
All question unigrams	88.4	78.2
Question bigrams	91.6	79.4
+informer q-grams	94.0	82.4
+informer hypernyms	**94.2**	**88.0**
Question unigrams + all informer	93.4	88.0
Only informer	92.2	85.0
Question bigrams + hypernyms	91.6	79.4

Table 2: Percent accuracy with linear SVMs, “perfect” informer spans, and various feature encodings.

Observe in Table 2 that the unigram baseline is already quite competitive with the best prior numbers, and exploiting perfect informer spans beats all known numbers. It is clear that both *informer q-grams* and *informer hypernyms* are very valuable features for question classification. The fact that no improvement was obtained with over *Question bigrams* using *Question hypernyms* highlights the importance of choosing a few relevant tokens as informers and designing suitable features on them.

Table 3 (columns b and e) shows the benefits from perfect informers broken down into broad question types. Questions with *what* as the trigger are the biggest beneficiaries, and they also form by far the most frequent category.

The remaining question, one that we address in the rest of the paper, is whether we can effectively and accurately automate the process of providing informer spans to the question classifier.

3.3 Informers provided by heuristics

In §4 we will propose a non-trivial solution to the informer-tagging problem. Before that, we must justify that such machinery is indeed required.

Some leading QA systems extract words very similar in function to informers from the parse tree of the question. Some (Singhal et al., 2000) pick the head of the first noun phrase detected by a shallow parser, while others use the head of the noun phrase adjoining the main verb (Ramakrishnan et al., 2004). Yet others (Harabagiu et al., 2000; Hovy et al., 2001) use hundreds of (unpublished to our knowledge) hand-built pattern-matching rules on the output of a full-scale parser.

A natural baseline is to use these extracted words, which we call “heuristic informers”, with an SVM just like we used “perfect” informers. All that remains is to make the heuristics precise.

How: For questions starting with *how*, we use the bigram starting with *how* unless the next word is a verb.

Wh: If the wh-word is not *how, what* or *which*, use the wh-word in the question as a separate feature.

WhNP: For questions having *what* and *which*, use the WHNP if it encloses a noun. WHNP is the Noun Phrase corresponding to the Wh-word, given by a sentence parser (see §4.2).

NP1: Otherwise, for *what* and *which* questions, the first (leftmost) noun phrase is added to yet another feature subspace.

Table 3 (columns c and f) shows that these already-messy heuristic informers do not capture the same signal quality as “perfect” informers. Our findings corroborate Li and Roth (2002), who report little benefit from adding head chunk features for the fine classification task.

Moreover, observe that using heuristic informer features *without* any word features leads to rather poor performance (column c), unlike using perfect informers (column b) or even CRF-predicted informer (column d, see §4). These clearly establish that the notion of an informer is nontrivial.

4 Using CRFs to label informers

Given informers are useful but nontrivial to recognize, the next natural question is, how can we learn to identify them automatically? From earlier sections, it is clear (and we give evidence later, see Table 5) that sequence and syntax information will be

		B	Only Informers			B+	B+	B+
Type	#Quest.	(Bigrams)	Perf.Inf	H.Inf	CRF.Inf	Perf.Inf	H.Inf	CRF.Inf
6 coarse classes								
what	349	**88.8**	89.4	69.6	79.3	**91.7**	87.4	91.4
which	11	**72.7**	100.0	45.4	81.8	**100.0**	63.6	81.8
when	28	100.0	100.0	100.0	100.0	100.0	100.0	100.0
where	27	100.0	96.3	100.0	96.3	100.0	100.0	100.0
who	47	100.0	100.0	100.0	100.0	100.0	100.0	100.0
how_*	32	100.0	96.9	100.0	100.0	100.0	100.0	100.0
rest	6	100.0	100.0	100.0	66.7	100.0	66.7	66.7
Total	500	91.6	92.2	*77.2*	84.6	94.2	*90.0*	93.4
50 fine classes								
what	349	**73.6**	82.2	61.9	78.0	**85.1**	79.1	83.1
which	11	**81.8**	90.9	45.4	73.1	**90.9**	54.5	81.8
when	28	100.0	100.0	100.0	100.0	100.0	100.0	100.0
where	27	92.6	85.2	92.6	88.9	88.9	92.5	88.9
who	47	97.9	93.6	93.6	93.6	100.0	100.0	97.9
how_*	32	87.5	84.3	81.2	78.1	87.5	90.6	90.6
rest	6	66.7	66.7	66.7	66.7	100.0	66.7	66.7
Total	500	79.4	85.0	*69.6*	78.0	88.0	*82.6*	86.2
		a	b	c	d	e	f	g

Table 3: Summary of % accuracy broken down by question type (referred from §3.2, §3.3 and §4.4). a: question bigrams, b: perfect informers only, c: heuristic informers only, d: CRF informers only, e–g: bigrams plus perfect, heuristic and CRF informers.

important.

We will model informer span identification as a sequence tagging problem. An automaton makes probabilistic transitions between hidden states y, one of which is an "informer generating state", and emits tokens x. We observe the tokens and have to guess which were produced from the "informer generating state".

Hidden Markov models are extremely popular for such applications, but recent work has shown that conditional random fields (CRFs) (Lafferty et al., 2001; Sha and Pereira, 2003) have a consistent advantage over traditional HMMs in the face of many redundant features. We refer the reader to the above references for a detailed treatment of CRFs. Here we will regard a CRF as largely a black box[3].

To train a CRF, we need a set of state nodes, a transition graph on these nodes, and tokenized text where each token is assigned a state. Once the CRF is trained, it can be applied to a token sequence, producing a predicted state sequence.

[3]We used `http://crf.sourceforge.net/`

4.1 State transition models

We started with the common 2-state "in/out" model used in information extraction, shown in the left half of Figure 2. State "1" is the informer-generating state. Either state can be initial and final (double circle) states.

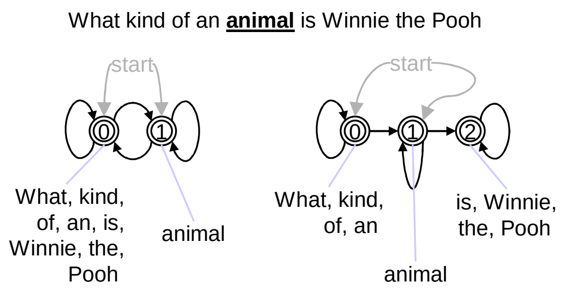

What kind of an **animal** is Winnie the Pooh
start
0
1
What, kind, of, an, is, Winnie, the, Pooh
animal
start
0
1
2
What, kind, of, an
is, Winnie, the, Pooh
animal

Figure 2: 2- and 3-state transition models.

The 2-state model can be myopic. Consider the question pair

A: What country is the largest producer of wheat?
B: Name the largest producer of wheat

The $i \pm 1$ context of *producer* is identical in A and B. In B, for want of a better informer, we would want *producer* to be flagged as the informer, although it might refer to a country, person, animal, company, etc. But in A, *country* is far more precise.

Any 2-state model that depends on positions $i \pm 1$ to define features will fail to distinguish between A and B, and might select both *country* and *producer* in A. As we have seen with heuristic informers, polluting the informer pool can significantly hurt SVM accuracy.

Therefore we also use the 3-state "begin/in/out" (BIO) model. The initial state cannot be "2" in the 3-state model; all states can be final. The 3-state model allows at most one informer span. Once the 3-state model chooses *country* as the informer, it is unlikely to stretch state 1 up to *producer*.

There is no natural significance to using four or more states. Besides, longer range syntax dependencies are already largely captured by the parser.

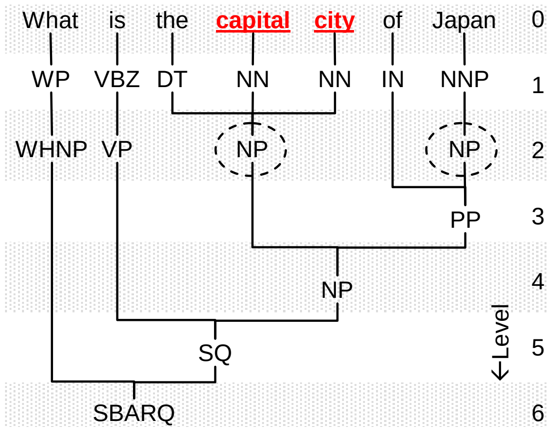


Figure 3: Stanford Parser output example.

4.2 Features from a parse of the question

Sentences with similar parse trees are likely to have the informer in similar positions. This was the intuition behind Zhang et al.'s tree kernel, and is also our starting point. We used the Stanford Lexicalized Parser (Klein and Manning, 2003) to parse the question. (We assume familiarity with parse tree notation for lack of space.) Figure 3 shows a sample parse tree organized in levels. Our first step was to translate the parse tree into an equivalent multi-resolution tabular format shown in Table 4.

i	1	2	3	4	5	6	7
y_i	0	0	0	1	1	2	2
x_i	What	is	the	capital	city	of	Japan
$\ell \downarrow$	Features for x_is						
1	WP,1	VBZ,1	DT,1	NN,1	NN,1	IN,1	NNP,1
2	WHNP,1	VP,1	NP,1	**NP**,1	NP,1	Null,1	**NP,2**
3	Null,1	Null,1	Null,1	Null,1	Null,1	PP,1	PP,1
4	Null,1	Null,1	NP,1	NP,1	NP,1	NP,1	NP,1
5	Null,1	SQ,1	SQ,1	SQ,1	SQ,1	SQ,1	SQ,1
6	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ

Table 4: A multi-resolution tabular view of the question parse showing `tag` and `num` attributes. *capital city* is the informer span with $y = 1$.

Cells and attributes: A labeled question comprises the token sequence $x_i; i = 1, \ldots$ and the label sequence $y_i, i = 1, \ldots$ Each x_i leads to a column vector of observations. Therefore we use matrix notation to write down x: A table cell is addressed as $x[i, \ell]$ where i is the token position (column index) and ℓ is the level or row index, 1–6 in this example. (Although the parse tree can be arbitrarily deep, we found that using features from up to level $\ell = 2$ was adequate.)

Intuitively, much of the information required for spotting an informer can be obtained from the part of speech of the tokens and phrase/clause attachment information. Conversely, specific word information is generally sparse and misleading; the same word may or may not be an informer depending on its position. E.g., "What birds eat snakes?" and "What snakes eat birds?" have the same words but different informers. Accordingly, we observe two properties at each cell:

`tag`: The syntactic class assigned to the cell by the parser, e.g. $x[4, 2]$`.tag` = `NP`. It is well-known that POS and chunk information are major clues to informer-tagging, specifically, informers are often nouns or noun phrases.

`num`: Many heuristics exploit the fact that the first NP is known to have a higher chance of containing informers than subsequent NPs. To capture this positional information, we define `num` of a cell at $[i, \ell]$ as one plus the number of distinct contiguous chunks to the left of $[i, \ell]$ with `tag`s equal to $x[4, 2]$`.tag`. E.g., at level 2 in the table above, *the capital city* forms the first NP, while *Japan* forms the second NP. Therefore $x[7,2]\texttt{.num} = 2$.

In conditional models, it is notationally convenient to express features as functions on (x_i, y_i). To one unfamiliar with CRFs, it may seem strange that y_i is passed as an argument to features. At training time, y_i is indeed known, and at testing time, the CRF algorithm efficiently finds the most probable sequence of y_is using a Viterbi search. True labels are not revealed to the CRF at testing time.

Cell features `IsTag` and `IsNum`: E.g., the observation "$y_4 = 1$ and $x[4,2]\texttt{.tag} = \texttt{NP}$" is captured by the statement that "position 4 fires the feature $\texttt{IsTag}_{1,\texttt{NP},2}$" (which has a boolean value). There is an $\texttt{IsTag}_{y,t,\ell}$ feature for each (y, t, ℓ) triplet. Similarly, for every possible state y, every possible `num` value n (up to some maximum horizon), and every level ℓ, we define boolean features $\texttt{IsNum}_{y,n,\ell}$. E.g., position 7 fires the feature $\texttt{IsNum}_{2,2,2}$ in the 3-state model, capturing the statement "$x[7,2]\texttt{.num} = 2$ and $y_7 = 2$".

Adjacent cell features `IsPrevTag` and `IsNextTag`: Context can be exploited by a CRF by coupling the state at position i with observations at positions adjacent to position i (extending to larger windows did not help). To capture this, we use more boolean features: position 4 fires the feature $\texttt{IsPrevTag}_{1,\texttt{DT},1}$ because $x[3,1]\texttt{.tag} = \texttt{DT}$ and $y_4 = 1$. Position 4 also fires $\texttt{IsPrevTag}_{1,\texttt{NP},2}$ because $x[3,2]\texttt{.tag} = \texttt{NP}$ and $y_4 = 1$. Similarly we define a $\texttt{IsNextTag}_{y,t,\ell}$ feature for each possible (y, t, ℓ) triple.

State transition features `IsEdge`: Position i fires feature $\texttt{IsEdge}_{u,v}$ if $y_{i-1} = u$ and $y_i = v$. There is one such feature for each state-pair (u, v) allowed by the transition graph. In addition we have sentinel features $\texttt{IsBegin}_u$ and $\texttt{IsEnd}_u$ marking the beginning and end of the token sequence.

4.3 Informer-tagging accuracy

We study the accuracy of our CRF-based informer tagger wrt human informer annotations. In the next section we will see the effect of CRF tagging on question classification.

There are at least two useful measures of informer-tagging accuracy. Each question has a known set I_k of informer tokens, and gets a set of tokens I_c flagged as informers by the CRF. For each question, we can grant ourself a reward of 1 if $I_c = I_k$, and 0 otherwise. In §3.1, informers were regarded as a separate (high-value) bag of words. Therefore, overlap between I_c and I_k would be a reasonable predictor of question classification accuracy. We use the Jaccard similarity $|I_k \cap I_c| / |I_k \cup I_c|$. Table 5 shows the effect of using diverse feature sets.

Features used	Fraction $I_c = I_k$	Jaccard overlap
IsTag	0.368	0.396
+IsNum	0.474	0.542
+IsPrevTag+IsNextTag	0.692	0.751
+IsEdge+IsBegin+IsEnd	**0.848**	**0.867**

Table 5: Effect of feature choices.

- `IsTag` features are not adequate.
- `IsNum` features improve accuracy 10–20%.
- `IsPrevTag` and `IsNextTag` ("+Prev +Next") add over 20% of accuracy.
- `IsEdge` transition features help exploit Markovian dependencies and adds another 10–15% accuracy, showing that sequential models are indeed required.

Type	#Quest.	Heuristic Informers	2-state CRF	3-state CRF
what	349	57.3	68.2	83.4
which	11	77.3	83.3	77.2
when	28	75.0	98.8	100.0
where	27	84.3	100.0	96.3
who	47	55.0	47.2	96.8
how_*	32	90.6	88.5	93.8
rest	6	66.7	66.7	77.8
Total	500	62.4	71.2	86.7

Table 6: Effect of number of CRF states, and comparison with the heuristic baseline (Jaccard accuracy expressed as %).

Table 6 shows that the 3-state CRF performs much better than the 2-state CRF, especially on difficult questions with *what* and *which*. It also compares the Jaccard accuracy of informers found by the CRF vs. informers found by the heuristics described in §3.3. Again we see a clear superiority of the CRF approach.

Unlike the heuristic approach, the CRF approach is relatively robust to the parser emitting a somewhat incorrect parse tree, which is not uncommon. The heuristic approach picks the “easy” informer, *who*, over the better one, *CEO*, in “Who is the CEO of IBM”. Its bias toward the NP-head can also be a problem, as in “What country’s *president* . . .”.

4.4 Question classification accuracy

We have already seen in §3.2 that perfect knowledge of informers can be a big help. Because the CRF can make mistakes, the margin may decrease. In this section we study this issue.

We used questions with human-tagged informers (§3.2) to train a CRF. The CRF was applied back on the training questions to get informer predictions, which were used to train the 1-vs-1 SVM meta-learner (§3). Using CRF-tagged and not human-tagged informers may seem odd, but this lets the SVM learn and work around systematic errors in CRF outputs.

Results are shown in columns d and g of Table 3. Despite the CRF tagger having about 15% error, we obtained 86.2% SVM accuracy which is rather close to the the SVM accuracy of 88% with perfect informers.

The CRF-generated tags, being on the training data, might be more accurate that would be for unseen test cases, potentially misleading the SVM. This turns out not to be a problem: clearly we are very close to the upper bound of 88%. In fact, anecdotal evidence suggests that using CRF-assigned tags actually helped the SVM.

5 Conclusion

We presented a new approach to inferring the type of the answer sought by a well-formed natural language question. We introduced the notion of a span of *informer tokens* and extract it using a sequential graphical model with a novel feature representation derived from the parse tree of the question. Our approach beats the accuracy of recent algorithms, even ones that used max-margin methods with sophisticated kernels defined on parse trees.

An intriguing feature of our approach is that when an informer (*actor*) is narrower than the question class (*person*), we can exploit direct hypernymy connections like *actor* to *Tom Hanks*, if available. Existing knowledge bases like WordNet and Wikipedia, combined with intense recent work (Etzioni et al., 2004) on bootstrapping is-a hierarchies, can thus lead to potentially large benefits.

Acknowledgments: Thanks to Sunita Sarawagi for help with CRFs, and the reviewers for improving the presentation.

References

P. K Chan and S. J Stolfo. 1993. Experiments in multistrategy learning by meta-learning. In *CIKM*, pages 314–323, Washington, DC.

S Dumais, M Banko, E Brill, J Lin, and A Ng. 2002. Web question answering: Is more always better? In *SIGIR*, pages 291–298.

O Etzioni, M Cafarella, et al. 2004. Web-scale information extraction in KnowItAll. In *WWW Conference*, New York. ACM.

K Hacioglu and W Ward. 2003. Question classification with support vector machines and error correcting codes. In *HLT*, pages 28–30.

S Harabagiu, D Moldovan, M Pasca, R Mihalcea, M Surdeanu, R Bunescu, R Girju, V Rus, and P Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *TREC 9*, pages 479–488. NIST.

E Hovy, L Gerber, U Hermjakob, M Junk, and C.-Y Lin. 2001. Question answering in Webclopedia. In *TREC 9*. NIST.

R Khardon, D Roth, and L. G Valiant. 1999. Relational learning for NLP using linear threshold elements. In *IJCAI*.

D Klein and C. D Manning. 2003. Accurate unlexicalized parsing. In *ACL*, volume 41, pages 423–430.

C Kwok, O Etzioni, and D. S Weld. 2001. Scaling question answering to the Web. In *WWW Conference*, volume 10, pages 150–161, Hong Kong.

J Lafferty, A McCallum, and F Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

X Li and D Roth. 2002. Learning question classifiers. In *COLING*, pages 556–562.

G Ramakrishnan, S Chakrabarti, D. A Paranjpe, and P Bhattacharyya. 2004. Is question answering an acquired skill? In *WWW Conference*, pages 111–120, New York.

F Sha and F Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*, pages 134–141.

A Singhal, S Abney, M Bacchiani, M Collins, D Hindle, and F Pereira. 2000. AT&T at TREC-8. In *TREC 8*, pages 317–330. NIST.

D Zhang and W Lee. 2003. Question classification using support vector machines. In *SIGIR*, pages 26–32.

Z Zheng. 2002. AnswerBus question answering system. In *HLT*.

A Practically Unsupervised Learning Method to Identify Single-Snippet Answers to Definition Questions on the Web

Ion Androutsopoulos and **Dimitrios Galanis**
Department of Informatics
Athens University of Economics and Business
Patission 76, GR-104 34, Athens, Greece

Abstract

We present a practically unsupervised learning method to produce single-snippet answers to definition questions in question answering systems that supplement Web search engines. The method exploits on-line encyclopedias and dictionaries to generate automatically an arbitrarily large number of positive and negative definition examples, which are then used to train an SVM to separate the two classes. We show experimentally that the proposed method is viable, that it outperforms the alternative of training the system on questions and news articles from TREC, and that it helps the search engine handle definition questions significantly better.

1 Introduction

Question answering (QA) systems for document collections typically aim to identify in the collections text snippets (e.g., 50 or 250 characters long) or exact answers (e.g., names, dates) that answer natural language questions submitted by their users. Although they are commonly evaluated on newspaper archives, as in the TREC QA track, QA systems can also supplement Web search engines, to help them return snippets, as opposed to Web pages, that provide more directly the information users require.

Most current QA systems first classify the input question into one of several categories (e.g., questions asking for locations, persons, etc.), producing expectations for types of named entities that must be present in the answer (locations, person names, etc.). Using the question's terms as a query, an information retrieval (IR) system identifies relevant documents. Snippets of these documents are then selected and ranked, using criteria such as whether or not they contain the expected types of named entities, the percentage of the question's terms they contain, etc. The system then outputs the most highly-ranked snippets, or named entities therein.

The approach highlighted above performs poorly with definition questions (e.g., "What is gasohol?", "Who was Duke Ellington?"), because definition questions do not generate expectations for particular types of named entities, and they typically contain only a single term. Definition questions are particularly common; in the QA track of TREC-2001, where the distribution of question types reflected that of real user logs, 27% of the questions were requests for definitions. Of course, answers to many definition questions can be found in on-line encyclopedias and dictionaries.[1] There are always, however, new or less widely used terms that are not included in such resources, and this is also true for many names of persons and products. Hence, techniques to discover definitions in ordinary Web pages and other document collections are valuable. Definitions of this kind are often 'hidden' in oblique contexts (e.g., "He said that *gasohol, a mixture of gasoline and ethanol*, has been great for his business.").

In recent work, Miliaraki and Androutsopoulos (2004), hereafter M&A, proposed a method we call

[1]See, for example, Wikipedia (http://www.wikipedia.org/). WordNet's glosses are another on-line source of definitions.

DEFQA, which handles definition questions. The method assumes that a question preprocessor separates definition from other types of questions, and that in definition questions this module also identifies the term to be defined, called the *target term*.[2] The input to DEFQA is a (possibly multi-word) target term, along with the r most highly ranked documents that an IR system returned for that term. The output is a list of k 250-character snippets from the r documents, at least one of which must contain an acceptable short definition of the target term, much as in the QA track of TREC-2000 and TREC-2001.[3]

We note that since 2003, TREC requires definition questions to be answered by lists of *complementary* snippets, jointly providing a range of information nuggets about the target term (Voorhees, 2003). In contrast, here we focus on locating single-snippet definitions. We believe this task is still interesting and of practical use. For example, a list of single-snippet definitions accompanied by their source URLs is a good starting point for users of search engines wishing to obtain definitions. Single-snippet definitions can also be useful in information extraction, where the templates to be filled in often require short entity descriptions. We also note that the post-2003 TREC task has encountered evaluation problems, because it is difficult to agree on which nuggets should be included in the multi-snippet definitions (Hildebrandt et al., 2004). In contrast, our experimental results of Section 4 indicate strong inter-assessor agreement for single-snippet answers, suggesting that it is easier to agree upon what constitutes an acceptable single-snippet definition.

DEFQA relies on an SVM, which is trained to classify 250-character snippets that have the target term at their centre, hereafter called *windows*, as acceptable definitions or non-definitions.[4] To train the SVM, a collection of q training target terms is used; M&A used the target terms of definition questions from TREC-2000 and TREC-2001. The terms are submitted to an IR system, which returns the r most highly ranked documents per target term. The windows of the $q \cdot r$ resulting documents are tagged as acceptable definitions or non-definitions, and become the training instances of the SVM. At run time, when a definition question is submitted, the r top-ranked documents are obtained, their windows are collected, and for each window the SVM returns a score indicating how confident it is that the window is a definition. The k windows with the highest confidence scores are then reported to the user.

The SVM actually operates on vector representations of the windows, that comprise the verdicts or attributes of previous methods by Joho and Sanderson (2000) and Prager et al. (2002), as well as attributes corresponding to automatically acquired lexical patterns. On TREC-2000 and TREC-2001 data, M&A found that DEFQA clearly outperformed the original methods of Joho and Sanderson and Prager et al. Their best configuration answered correctly 73% of 160 definition questions in a cross-validation experiment with $k = 5$, $r = 50$, $q = 160$.

A limitation of DEFQA is that it cannot be trained easily on new document collections, because it requires the training windows to be tagged as definitions or non-definitions. In the experiments of M&A, there were 18,473 training windows. Tagging them was easy, because the windows were obtained from TREC questions and documents, and the TREC organizers provide Perl patterns that can be used to judge whether a snippet from TREC's documents is among the acceptable answers of a TREC question.[5] For non-TREC questions and document collections, however, where such patterns are unavailable, separating thousands of training windows into the two categories by hand is a laborious task.

In this paper, we consider the case where DEFQA is used as an add-on to a Web search engine. There are three training alternatives in this setting: (i) train DEFQA on TREC questions and documents; (ii) train DEFQA on a large collection of manually tagged training windows obtained from Web pages that the search engine returned for training target terms; or (iii) devise techniques to tag automatically the training windows of (ii). We have developed a technique along alternative (iii), which exploits on-

[2]Alternatively, the user can be asked to specify explicitly the question type and target term via a form-based interface.

[3]Definition questions were not considered in TREC-2002.

[4]See, for example, Scholkopf and Smola (2002) for information on SVMs. Following M&A, we use a linear SVM, as implemented by Weka's SMO class (http://www.cs.waikato.ac.nz/ml/weka/). The windows may be shorter than 250 characters, when the surrounding text is limited.

[5]The patterns' judgements are not always perfect, which introduces some noise in the training examples.

line encyclopedias and dictionaries. This allows us to generate and tag automatically an arbitrarily large number of training windows, in effect converting DEFQA to an unsupervised method. We show experimentally that the new unsupervised method is viable, that it outperforms alternative (i), and that it helps the search engine handle definition questions significantly better than on its own.

2 Attributes of DEFQA

DEFQA represents each window as a vector comprising the values of the following attributes:[6]

SN: The ordinal number of the window in the document, in our case Web page, it originates from. The intuition is that windows that mention the target term first in a document are more likely to define it.

WC: What percentage of the 20 words that are most frequent across all the windows of the target term are present in the particular window represented by the vector. A stop-list and a stemmer are applied first when computing WC.[7] In effect, the 20 most frequent words form a simplistic centroid of all the candidate answers, and WC measures how close the vector's window is to this centroid.

RK: The ranking of the Web page the window originates from, as returned by the search engine.

Manual patterns: 13 binary attributes, each signalling whether or not the window matches a different manually constructed lexical pattern (e.g., "*target*, a/an/the", as in "Tony Blair, the British prime minister"). The patterns are those used by Joho and Sanderson, and four more added by M&A. They are intended to perform well across text genres.

Automatic patterns: A collection of m binary attributes, each showing if the window matches a different automatically acquired lexical pattern. The patterns are sequences of n tokens ($n \in \{1, 2, 3\}$) that must occur either directly before or directly after the target term (e.g., "*target*, which is"). These patterns are acquired as follows. First, all the n-grams that occur directly before or after the target terms in the training windows are collected. The n-grams that have been encountered at least 10 times are considered candidate patterns. From those, the m patterns with the highest precision scores are retained, where *precision* is the number of training definition windows the pattern matches divided by the total number of training windows it matches. We set m to 200, the value that led to the best results in the experiments of M&A. The automatically acquired patterns allow the system to detect definition contexts that are not captured by the manual patterns, including genre-specific contexts.

M&A also explored an additional attribute, which carried the verdict of Prager et al.'s WordNet-based method (2002). However, they found the additional attribute to lead to no significant improvements, and, hence, we do not use it. We have made no attempt to extend the attribute set of M&A; for example, with attributes showing if the window contains the target term in italics, if the window is part of a list that looks like a glossary, or if the window derives from an authority Web page. We leave such extensions for future work. Our contribution is the automatic generation of training examples.

3 Generating training examples

When training DEFQA on windows from Web pages, a mechanism to tag the training windows as definitions or non-definitions is required. Rather than tagging them manually, we use a measure of how similar the wording of each training window is to the wording of definitions of the same target term obtained from on-line encyclopedias and dictionaries. This is possible because we pick training target terms for which there are several definitions in different on-line encyclopedias and dictionaries; hereafter we call these *encyclopedia definitions*.[8] Training windows whose wording is very similar to that of the corresponding encyclopedia definitions are tagged as definition windows (*positive examples*), while windows whose wording differs significantly from the encyclopedia definitions are tagged as non-definitions (*negative examples*). Training windows for which the similarity score does not indicate great similarity or dissimilarity to the wording of the encyclopedia definitions are excluded from DEFQA's

[6] SN and WC originate from Joho and Sanderson (2000).

[7] We use the 100 most frequent words of the British National Corpus as the stop-list, and Porter's stemmer.

[8] We use randomly selected entries from the index of http://www.encyclopedia.com/ as training terms, and Google's 'define:' feature, that returns definitions from on-line encyclopedias and dictionaries, to obtain the encyclopedia definitions.

training, as they cannot be tagged as positive or negative examples with sufficiently high confidence.

Note that encyclopedia definitions are used only to tag training windows. Once the system has been trained, it can be used to discover on ordinary Web pages definitions of terms for which there are no encyclopedia definitions, and indeed this is the main purpose of the system. Note also that we train DEFQA on windows obtained from Web pages returned by the search engine for training terms. This allows it to learn characteristics of the particular search engine being used; for example, what weight to assign to RK, depending on how much the search engine succeeds in ranking pages containing definitions higher. More importantly, it allows DEFQA to select lexical patterns that are indicative of definitions in Web pages, as opposed to patterns that are indicative of definitions in electronic encyclopedias and dictionaries. The latter explains why we do not train DEFQA directly on encyclopedia definitions; another reason is that DEFQA requires both positive and negative examples, while encyclopedia definitions provide only positive ones.

We now explain how we compute the similarity between a training window and the collection C of encyclopedia definitions for the window's target term. We first remove stop-words, punctuation, other non-alphanumeric characters and the target term from the training window, and apply a stemmer, leading to a new form W of the training window. We then compute the similarity of W to C as:

$$sim(W, C) = 1/|W| \cdot \Sigma_{i=1}^{|W|} sim(w_i, C)$$

where $|W|$ is the number of distinct words in W, and $sim(w_i, C)$ is the similarity of the i-th distinct word of W to C, defined as follows:

$$sim(w_i, C) = fdef(w_i, C) \cdot idf(w_i)$$

$fdef(w_i, C)$ is the percentage of definitions in C that contain w_i, and $idf(w_i)$ is the inverse document frequency of w_i in the British National Corpus (BNC):

$$idf(w_i) = 1 + log \frac{N}{df(w_i)}$$

N is the number of documents in BNC, and $df(w_i)$ the number of BNC documents where w_i occurs; if w_i does not occur in BNC, we use the lowest df score of BNC. $sim(w_i, C)$ is highest for words that occur in all the encyclopedia definitions and are used rarely in English. A training window with a large proportion of such words most probably defines the target term. More formally, given two thresholds t_+ and t_- with $t_- \leq t_+$, we tag W as a positive example if $sim(W, C) \geq t_+$, as a negative example if $sim(W, C) \leq t_-$, and we exclude it from the training of DEFQA if $t_- < sim(W, C) < t_+$. Hereafter, we refer to this method of generating training examples as the *similarity method*.

To select reasonable values for t_+ and t_-, we conducted a preliminary experiment for $t_- = t_+ = t$; i.e., both thresholds were set to the same value t and no training windows were excluded. We used $q = 130$ training target terms from TREC definition questions, for which we had multiple encyclopedia definitions. For each term, we collected the $r = 10$ most highly ranked Web pages.[9] To alleviate the class imbalance problem, whereby the positive examples (definitions) are much fewer than the negative ones (non-definitions), we kept only the first 5 windows from each Web page ($SN \leq 5$), based on the observation that windows with great SN scores are almost certainly non-definitions; we do the same in the training stage of all the experiments of this paper, and at run-time, when looking for windows to report, we ignore windows with $SN > 5$. From the resulting collection of training windows, we selected randomly 400 windows, and tagged them both manually and via the similarity method, with t ranging from 0 to 1. Figures 1 and 2 show the precision and recall of the similarity method on positive and negative training windows, respectively, for varying t. Here, *positive precision* is the percentage of training windows the similarity method tagged as positive examples (definitions) that were indeed positive; the true classes of the training windows were taken to be those assigned by the human annotators. *Positive recall* is the percentage of truly positive examples that the similarity method tagged as positive. *Negative precision* and *negative recall* are defined similarly.

Figures 1 and 2 indicate that there is no single threshold t that achieves both high positive precision and high negative precision. To be confident

[9] In all our experiments, we used the Altavista search engine.

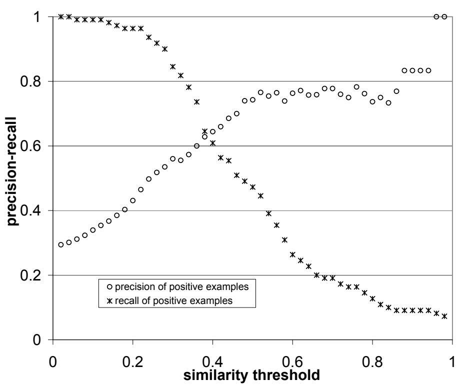


Figure 1: Positive precision and recall

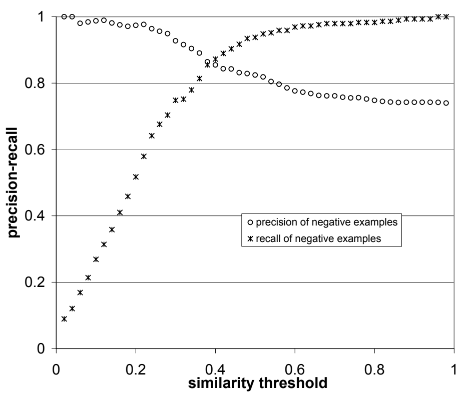


Figure 2: Negative precision and recall

that the training windows the similarity method will tag as positive examples are indeed positive (high positive precision), one has to set t close to 1; and to be confident that the training windows the similarity method will tag as negative examples are indeed negative (high negative precision), t has to be set close to 0. This is why we use two separate thresholds and discard the training windows whose similarity score is between t_- and t_+. Figures 1 and 2 also indicate that in both positive and negative examples the similarity method achieves perfect precision only at the cost of very low recall; i.e., if we insist that all the resulting training examples must have been tagged correctly (perfect positive and negative precision), the resulting examples will be very few (low positive and negative recall). There is also another consideration when selecting t_- and t_+: the ratio of positive to negative examples that the similarity method generates must be approximately the same as the true ratio before discarding any training windows, in order to avoid introducing an artificial bias in the training of DEFQA's SVM; the true ratio among the 400 training windows before discarding any windows was approximately $0.37 : 1$.

Based on the considerations above, in the remaining experiments of this paper we set t_+ to 0.5. In Figure 1, this leads to a positive precision of 0.72 (and positive recall 0.49), which does not improve much by adopting a larger t_+, unless one is willing to set t_+ at almost 1 at the price of very low positive recall. In the case of t_-, setting it to any value less than 0.34 leads to a negative precision above 0.9, though negative recall drops sharply as t_- approaches 0 (Figure 2). For example, setting t_- to 0.32, leads to 0.92 negative precision, 0.75 negative recall, and approximately the same positive to negative ratio ($0.31 : 1$) as the true observed ratio. In the experiments of Section 4, we keep t_+ fixed to 0.5, and set t_- to the value in the range $(0, 0.34)$ that leads to the positive to negative ratio that is closest to the true ratio we observed in the 400 windows.

The high negative precision we achieve (> 0.9) suggests that the resulting negative examples are almost always truly negative. In contrast, the lower positive precision (0.72) indicates that almost one in every four resulting positive examples is in reality a non-definition. This is a point where our similarity method needs to be improved; we return to this point in Section 6. Our experiments, however, show that despite this noise, the similarity method already outperforms the alternative of training DEFQA on TREC data. Note also that once the thresholds have been selected, we can generate automatically an arbitrarily large set of training examples, by starting with a sufficiently large number q of training terms to compensate for discarded training examples.

4 Evaluation

We tested two different forms of DEFQA. The first one, dubbed DEFQA^t, was trained on the $q = 160$ definition questions of TREC-2000 and TREC-2001 and the corresponding TREC documents, resulting in 3,800 training windows.[10] The second form of DE-

[10]For each question, the TREC organizers provide the 50 most highly ranked documents that an IR engine returned from the TREC document collection. We keep the top $r = 10$ of these documents, while M&A kept all 50. Furthermore, as discussed in Section 3, we retain up to the first 5 windows from each doc-

FQA, dubbed DEFQA^s, was trained via the similarity method, with $q = 480$ training target terms, leading to 7,200 training windows; as discussed in Section 3, one of the advantages of the similarity method is that one can generate an arbitrarily large set of training windows. As in the preliminary experiment of Section 3, r (Web pages per target term) was set to 10 in both systems. To simplify the evaluation and test DEFQA in a more demanding scenario, we set k to 1, i.e., the systems were allowed to return only one snippet per question, as opposed to the more lenient $k = 5$ in the experiments of M&A.

We also wanted a measure of how well DEFQA^t and DEFQA^s perform compared to a search engine on its own. For this purpose, we compared the performance of the two systems to that of a baseline, dubbed BASE^1, which always returns the first window of the Web page the search engine ranked first. In a search engine that highlights question terms in the returned documents, the snippet returned by BASE^1 is presumably the first snippet a user would read hoping to find an acceptable definition. To study how much DEFQA^t and DEFQA^s improve upon random behaviour, we also compared them to a second baseline, BASE^r, which returns a randomly selected window among the first five windows of all r Web pages returned by the search engine.

All four systems were evaluated on 81 unseen target terms. Their responses were judged independently by two human assessors, who had to mark each response as containing an acceptable short definition or not. As already pointed out, DEFQA^t and DEFQA^s consult encyclopedia definitions only during training, and at run time the systems are intended to be used with terms for which no encyclopedia definitions are available. During this evaluation, however, we deliberately chose the 81 test terms from the index of an on-line encyclopedia. This allowed us to give the encyclopedia's definitions to the assessors, to help them judge the acceptability of the single-snippet definitions the systems located on Web pages; many terms where related to, for example, medicine or biology, and without the encyclopedia's definitions the assessors would not be aware of their meanings. The following is a snippet returned correctly by DEFQA^s for 'genome':

discipline comparative genomics functional genomics bioinformatics the emergence of genomics as a discipline in 1920 , the term genome was proposed to denote the totality of all genes on all chromosomes in the nucleus of a cell . biology has...

while what follows is a non-definition snippet returned wrongly by BASE^1:

what is a genome national center for biotechnology information about ncbi ncbi at a glance a science primer databases...

The examples illustrate the nature of the snippets that the systems and assessors had to consider. The snippets often contain phrases that acted as links in the original pages, or even pieces of programming scripts that our rudimental preprocessing failed to remove. (We remove only HTML tags, and apply a simplistic tokenizer.) Nevertheless, in most cases the assessors had no trouble agreeing whether or not the resulting snippets contained acceptable short definitions. K_{Co} was 0.80, 0.81, 0.90, 0.89, and 0.86 in the assessment of the responses of DEFQA^s, DEFQA^t, BASE^r, BASE^1, and all responses, respectively, indicating strong inter-assessor agreement.[11] The agreement was slightly lower in DEFQA^s and DEFQA^t, because there were a few marginally acceptable or truncated definitions the assessors were uncertain about. There were also 4 DEFQA^s answers and 3 BASE^1 answers that defined secondary meanings of the target terms; e.g., apart from a kind of lizard, 'gecko' is also the name of a graphics engine, and 'Exodus' is also a programme for ex-offenders. Such answers were counted as wrong, though this may be too strict. With a larger k, there would be space to return both the main and secondary meanings, and the evaluation could require this.

Table 1 shows that DEFQA^s answered correctly approximately 6 out of 10 definition questions. This is lower than the score reported by M&A (73%), but remarkably high given that in our evaluation the systems were allowed to return only one snippet per question; i.e., the task was much harder than in M&A's experiments. DEFQA^s answered correctly more than twice as many questions as DEFQA^t, despite the fact that its training data contained a lot of noise. (Single-tailed difference-of-proportions tests show that all the differences of Table 1 are statisti-

ument. This is why we have fewer training windows than M&A.

[11]We follow the notation of Di Eugenio and Glass (2004). The $K_{S\&C}$ figures were identical. The $2 \cdot P(A) - 1$ figures were 0.80, 0.85, 0.95, 0.95, and 0.89 respectively.

	assessor 1	assessor 2	average
BASE^r	14.81 (12)	14.81 (12)	14.81 (12)
BASE^1	14.81 (12)	12.35 (10)	13.58 (11)
DEFQA^t	25.93 (21)	25.93 (21)	25.93 (21)
DEFQA^s	55.56 (45)	60.49 (49)	58.02 (47)

Table 1: Percentage of questions answered correctly

cally significant at $\alpha = 0.001$.) The superiority of DEFQA^s appears to be mostly due to its automatically acquired patterns. DEFQA^t too was able to acquire several good patterns (e.g., 'by *target*', 'known as *target*', '*target*, which is', '*target* is used in'), but its pattern set also comprises a large number of irrelevant n-grams; this had also been observed by M&A. In contrast, the acquired pattern set of DEFQA^s is much cleaner, with much fewer irrelevant n-grams, which is probably due to the largest, almost double, number of training windows. Furthermore, the pattern set of DEFQA^s contains many n-grams that are indicative of definitions on the Web. For example, many Web pages that define terms contain text of the form "What is a *target*? A *target* is…", and DEFQA^s has discovered patterns of the form 'what is a/an/the *target*', '? A/an/the *target*', etc. It has also discovered patterns like 'FAQ *target*', 'home page *target*', '*target* page' etc., that seem to be good indications of Web windows containing definitions.

Overall, DEFQA's process of acquiring lexical patterns worked better in DEFQA^s than in DEFQA^t, and we believe that the performance of DEFQA^s could be improved further by acquiring more than 200 patterns; we hope to investigate this in future work, along with an investigation of how the performance of DEFQA^s relates to q, the number of training target terms. Finally, note that the scores of both baselines are very poor, indicating that DEFQA^s performs significantly better than picking the first, or a random snippet among those returned by the search engine.

5 Related work

Definition questions have recently attracted several QA researchers. Many of the proposed approaches, however, rely on manually crafted patterns or heuristics to identify definitions, and do not employ learning algorithms (Liu et al., 2003; Fujii and Ishikawa, 2004; Hildebrandt et al., 2004; Xu et al., 2004).

Ng et al. (2001) use machine learning (C5 with boosting) to classify and rank candidate answers in a general QA system, but they do not treat definition questions in any special way; consequently, their worst results are for "What…?" questions, that presumably include definition questions. Ittycheriah and Roukos (2002) employ a maximum entropy model to rank candidate answers in a general-purpose QA system. Their maximum entropy model uses a very rich set of attributes, that includes 8,500 n-gram patterns. Unlike our work, their n-grams are five or more words long, they are coupled to two-word question prefixes, and, in the case of definition questions, they do not need to be anchored at the target term. The authors, however, do not provide separate performance figures for definition questions.

Blair-Goldensohn et al. (2003) focus on definition questions, but aim at producing coherent multi-snippet definitions, rather than single-snippet definitions. The heart of their approach is a component that uses machine learning (Ripper) to identify sentences that can be included in the multi-sentence definition. This component plays a role similar to that of our SVM, but it is intended to admit a larger range of sentences, and appears to employ only attributes conveying the ordinal number of the sentence in its document and the frequency of the target term in the sentence's context.

Since TREC-2003, several researchers have proposed ways to generate multi-snippet definitions (Cui et al., 2004; Fujii and Ishikawa, 2004; Hildebrandt et al., 2004; Xu et al., 2004). The typical approach is to locate definition snippets, much as in our work, and then report definition snippets that are sufficiently different; most of the proposals use some form of clustering to avoid reporting redundant snippets. Such methods could also be applied to DEFQA, to extend it to the post-2003 TREC task.

On-line encyclopedias and dictionaries have been used to handle definition questions in the past, but not as in our work. Hildebrandt et al. (2004) look up target terms in encyclopedias and dictionaries, and then, knowing the answers, try to find supporting evidence for them in the TREC document collection. Xu et al. (2004) collect from on-line encyclopedias and dictionaries words that co-occur with the target term; these words and their frequencies are then used as a centroid of the target term, and candidate answers are ranked by computing their similarity to the centroid. This is similar to our WC attribute.

Cui et al. (2004) also employ a form of centroid, comprising words that co-occur with the target term. The similarity to the centroid is taken into consideration when ranking candidate answers, but it is also used to generate training examples for a learning component that produces soft patterns, in the same way that we use the similarity method to produce training examples for the SVM. As in our work, the training examples that the centroid generates may be noisy, but the component that produces soft patterns manages to generalize over the noise. To the best of our knowledge, this is the only other unsupervised learning approach for definition questions that has been proposed. We hope to compare the two approaches experimentally in future work. For the moment, we can only point out that Cui et al.'s centroid approach generates only positive examples, while our similarity method generates both positive and negative ones; this allows us to use a principled SVM learner, as opposed to Cui et al.'s more ad hoc soft patterns that incorporate only positive examples.

6 Conclusions and future work

We presented an unsupervised method to learn to locate single-snippet answers to definition questions in QA systems that supplement Web search engines. The method exploits on-line encyclopedias and dictionaries to generate automatically an arbitrarily large number of positive and negative definition examples, which are then used to train an SVM to separate the two classes. We have shown experimentally that the proposed method is viable, that it outperforms training the QA system on TREC data, and that it helps the search engine handle definition questions significantly better than on its own.

We have already pointed out the need to improve the positive precision of the training examples. One way may be to combine our similarity method with Cui et al.'s centroids. We also plan to study the effect of including more automatically acquired patterns and using more training target terms. Finally, our method can be improved by including attributes for the layout and authority of Web pages.

References

S. Blair-Goldensohn, K.R. McKeown, and A.H. Schlaikjer. 2003. A hybrid approach for answering definitional questions. Technical Report CUCS-006-03, Columbia University.

H. Cui, M.-Y. Kan, and T.-S. Chua. 2004. Unsupervised learning of soft patterns for generating definitions from online news. In *Proceedings of WWW-2004*, pages 90–99, New York, NY.

B. Di Eugenio and M. Glass. 2004. The kappa statistic: A second look. *Comput. Linguistics*, 30(1):95–101.

A. Fujii and T. Ishikawa. 2004. Summarizing encyclopedic term descriptions on the Web. In *Proceedings of COLING-2004*, pages 645–651, Geneva, Switzerland.

W. Hildebrandt, B. Katz, and J. Lin. 2004. Answering definition questions using multiple knowledge sources. In *Proceedings of HLT-NAACL 2004*, pages 49–56, Boston, MA.

A. Ittycheriah and S. Roukos. 2002. IBM's statistical question answering system – TREC-11. In *Proceedings of TREC-2002*.

H. Joho and M. Sanderson. 2000. Retrieving descriptive phrases from large amounts of free text. In *Proc. of the 9th ACM Conference on Information and Knowledge Management*, pages 180–186, McLean, VA.

B. Liu, C.W. Chin, and H.T. Ng. 2003. Mining topic-specific concepts and definitions on the Web. In *Proceedings of WWW-2003*, Budapest, Hungary.

S. Miliaraki and I. Androutsopoulos. 2004. Learning to identify single-snippet answers to definition questions. In *Proceedings of COLING-2004*, pages 1360–1366, Geneva, Switzerland.

H.T. Ng, J.L.P. Kwan, and Y. Xia. 2001. Question answering using a large text database: A machine learning approach. In *Proceedings of EMNLP-2001*, pages 67–73, Pittsburgh, PA.

J. Prager, J. Chu-Carroll, and K. Czuba. 2002. Use of WordNet hypernyms for answering what-is questions. In *Proceedings of TREC-2001*.

B. Scholkopf and A. Smola. 2002. *Learning with kernels*. MIT Press.

E.M. Voorhees. 2003. Evaluating answers to definition questions. In *Proceedings of HLT-NAACL 2003*, pages 109–111, Edmonton, Canada.

J. Xu, R. Weischedel, and A. Licuanan. 2004. Evaluation of an extraction-based approach to answering definitional questions. In *Proceedings of SIGIR-2004*, pages 418–424, Sheffield, U.K.

Collective Content Selection for Concept-To-Text Generation

Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
regina@csail.mit.edu

Mirella Lapata
School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

Abstract

A content selection component determines which information should be conveyed in the output of a natural language generation system. We present an efficient method for automatically learning content selection rules from a corpus and its related database. Our modeling framework treats content selection as a collective classification problem, thus allowing us to capture contextual dependencies between input items. Experiments in a sports domain demonstrate that this approach achieves a substantial improvement over context-agnostic methods.

1 Introduction

Content selection is a fundamental task in concept-to-text generation (Reiter and Dale, 2000). A practical generation system typically operates over a large database with multiple entries that could potentially be included in a text. A content selection component determines what subset of this information to include in the generated document.

For example, consider the task of automatically generating game summaries, given a database containing statistics on Americal football. Table 1 shows an excerpt from such a database, and its corresponding game summary written by a journalist. A single football game is typically documented in hundreds of database entries — all actions, player positions, and scores are recorded, along with a wide range of comparative and aggregate statistics. Only a small fraction of this information is featured in a game summary. The content selection component aims to identify this subset.[1]

In existing generation systems the content selection component is manually crafted. Specifying content selection rules is, however, notoriously difficult, prohibitively so in large domains. It involves the analysis of a large number of texts from a domain-relevant corpus, familiarity with the associated database, and consultation with domain experts. Moreover, the task must be repeated for each domain anew.

This paper proposes a data-driven method for learning the content-selection component for a concept-to-text generation system. We assume that the learning algorithm is provided with a parallel corpus of documents and a corresponding database, in which database entries that should appear in documents are marked.

One possible approach is to formulate content selection as a standard binary classification task: predict whether an item is to be included on the basis of its attributes alone. In fact, this method is commonly used for content selection in text summarization (e.g., Kupiec et al., 1995). However, by treating each instance in isolation, we cannot guarantee that the selected database entries are related in a meaningful way, which is essential for the generation of a coherent text.

Rather than selecting each item separately, we propose a method for *collective content selection*, where all candidates are considered simultaneously for selection. Collective selection thereby allows us to explicitly optimize *coherence* in the generated

[1]The organization of the selected information and its surface realization is typically handled by other components of the generation system, which are outside the scope of this paper.

Passing					
PLAYER	CP/AT	YDS	AVG	TD	INT
Brunell	17/38	192	6.0	0	0
Garcia	14/21	195	9.3	1	0
...	...	...	...	...	...

Rushing					
PLAYER	REC	YDS	AVG	LG	TD
Suggs	22	82	3.7	25	1
...	...	...	...	...	...

Fumbles				
PLAYER	FUM	LOST	REC	YDS
Coles	1	1	0	0
Portis	1	1	0	0
Davis	0	0	1	0
Little	0	0	1	0
...	...	...	...	...

Suggs rushed for 82 yards and scored a touchdown in the fourth quarter, leading the Browns to a 17-13 win over the Washington Redskins on Sunday. **Jeff Garcia went 14-of-21 for 195 yards and a TD** for the Browns, who didn't secure the win until **Coles fumbled** with 2:08 left. The Redskins (1-3) can pin their third straight loss on going just 1-for-11 on third downs, mental mistakes and **a costly fumble by Clinton Portis**. **Brunell finished 17-of-38 for 192 yards**, but was unable to get into any rhythm because Cleveland's defense shut down Portis. The Browns faked a field goal, but holder Derrick Frost was stopped short of a first down. **Brunell then completed a 13-yard pass to Coles, who fumbled** as he was being taken down and Browns safety Earl Little recovered.

Table 1: Sample target game description and example of database entries; boldface indicates correspondences between the text and the database (CP/AT: completed out of attempted, YDS: yards, AVG: average, TD: touchdown, INT: interception, REC: received, LG: longest gain, FUM: fumble).

text: semantically related entries are often selected together. In essence, the algorithm seeks a subset of candidates that is consistent with the individual preferences of each candidate, and at the same time maximally satisfies contextual constraints. A graph-based formulation of this optimization problem allows us to find an exact, globally optimal solution, using a min-cut algorithm.

Collective content selection is particularly beneficial to generation systems that operate over relational databases. Rich structural information available in a database can be readily utilized to determine semantic relatedness between different database entries. For instance, we can easily find all actions (e.g., touchdowns and fumbles) associated with a specific player in a game, which could be relevant for generating a summary centered around an individual. We show how to utilize database relations for discovering meaningful contextual links between database entries.

We evaluate our collective content selection model in a sports domain. The proposed content selection component operates over a large database containing descriptive statistics about American football games. Our model yields a 10% increase in F-score, when compared to a standard classification approach, thus demonstrating the benefits of collective content selection on this complex domain. Furthermore, our results empirically confirm the contribution of discourse constraints for content selection.

In the following section, we provide an overview of existing work on content selection. Then, we define the learning task and introduce our approach for collective content selection. Next, we present our experimental framework and data. We conclude the paper by presenting and discussing our results.

2 Related Work

The generation literature provides multiple examples of content selection components developed for various domains (Kukich, 1983; McKeown, 1985; Sripada et al., 2001; Reiter and Dale, 2000). A common theme across different approaches is the emphasis on coherence: related information is selected "to produce a text that hangs together" (McKeown, 1985). Similarly, our method is also guided by coherence constraints. In our case these constraints are derived automatically, while in symbolic generation systems coherence is enforced by analyzing a large number of texts from a domain-relevant corpus and careful hand-crafting of content selection rules.

Duboue and McKeown (2003) were the first to propose a method for learning content selection rules automatically, thus going beyond mere corpus analysis. They treat content selection as a classification task. Given a collection of texts associated with a domain-specific database, their model learns whether a database entry should be selected for presentation or not. Their modeling approach uses an expressive feature space while considering database entries in isolation.

Similarly to Duboue and McKeown (2003), we view content selection as a classification task and learn selection rules from a database and its corresponding corpus. In contrast to them, we consider all database entries simultaneously, seeking a globally optimal selection. Thus, we avoid the need for extensive feature engineering by incorporating discourse constraints into the learning framework. In addition, we assess whether data-driven methods for content selection scale up to large databases with thousands of interrelated entries, by evaluating our model in a sports domain. Previous work (Duboue and McKeown, 2003) has tackled the content selection problem for biographical summaries, a simpler domain with fewer entities and interactions among them.

3 The Task

We assume that the content selection component takes as input a set of database entries.[2] Each entry has a type and a set of attributes associated with its type. For instance, the database shown in Table 1 contains entries of three types — `Passing`, `Rushing` and `Fumbles`. Two entries are of type `Passing`, and each of them has six attributes — `PLAYER`, `CP/AT`, `YDS`, `AVG`, `TD`, `INT`. In addition, each entry has a label that specifies whether it should be included in a generated text or not.

During the training process, the learning algorithm is provided with n sets of database entries, each associated with a label whose value is known. In practice, we only require a parallel corpus of game summaries and database entries — label values are derived automatically via alignment (see Section 4 for more details).

[2]A terminological note: a database entry is analogous to a row in a relational table; throughout this paper we use the terms entity and database entry interchangeably.

The goal of the content selection component is to select entries from a database, i.e., to determine whether their label values are 0 or 1. Under this formulation, content selection is restricted to information available in the database; there is no attempt to induce new facts through inference.

In the next section, we describe our learning framework, and explain how it is applied to the content selection task.

3.1 The Collective Classification Approach

Generation of a coherent text crucially depends on our ability to select entities that are related in a meaningful way (McKeown, 1985). A content selection component that considers every entity in isolation does not have any means to enforce this important discourse constraint. We therefore formulate content selection as a *collective classification* task, where all entities that belong to the same database (i.e., the same football game) are considered simultaneously. This framework thus enables us to enforce contextual constraints by selecting related entities.

When considered in isolation, some database entries are more likely to be selected than others. In the American football domain, for example, entries of type `Rushing` are often extracted if they yield a touchdown.[3] Other `Rushing` entries (e.g., which do not deliver scoring points) are typically omitted. In general, the attributes of an entry can provide useful cues for predicting whether it should be selected. Therefore, we can perform content selection by applying a standard classifier on each entry. In Section 3.2, we explain in more detail how such a classifier can be trained.

We can also decide about entity selection by analyzing how entities relate to each other in the database. For instance, in a game where both quarterbacks[4] score, it is fairly unorthodox to mention the passing statistics for only one of them. Label assignments in which either both quarterbacks are selected, or both of them are omitted should be there-

[3]A touchdown is the primary method of scoring in American football; a touchdown is worth six points and is accomplished by gaining legal possession of the ball in the opponent's end zone.

[4]A quarterback in American football is the leader of a team's offense. In most offenses his primary duty is passing the ball. Quarterbacks are typically evaluated on their passing statistics, including total yardage, completion ratio, touchdowns, and the ability to avoid interceptions.

fore preferred. This relation between quarterback passing statistics exemplifies one type of link that can hold between entities. Other link types may encode contextual constraints, for instance capturing temporal and locational information. (In Section 3.3, we describe a method for discovering link types which encapsulate meaningful contextual dependencies.) By taking into account links between related entities, a content selection component can enforce dependencies in the labeling of related entities.

Our goal is to select a subset of database entities that maximally satisfies linking constraints and is as consistent as possible with the individual preferences of each entity. Thus, content selection can be naturally stated as an optimization problem — we wish to find a label assignment that *minimizes* the cost of violating the above constraints.

Let C_+ and C_- be a set of selected and omitted entities, respectively; $ind_+(x)$ and $ind_-(x)$ are scores that capture the individual preference of x to be either selected or omitted, and $link_L(x,y)$ reflects the degree of dependence between the labels of x and y based on a link of type L. Thus, the optimal label assignment for database entries $x_1,\ldots,x_n$ will minimize:

$$\sum_{x \in C_+} ind_-(x) + \sum_{x \in C_-} ind_+(x) + \sum_L \sum_{\substack{x_i \in C_+ \\ x_j \in C_-}} link_L(x_i,x_j)$$

The first two elements in this expression capture the penalty for assigning entities to classes against their individual preferences. For instance, the penalty for selecting an entry $x \in C_+$ will equal $ind_-(x)$, i.e., x's individual preference of being ommitted. The third term captures a linking penalty for all pairs of entities (x_i,x_j) that are connected by a link of type L, and are assigned to different classes.

This formulation is similar to the energy minimization framework, which is commonly used in image analysis (Besag, 1986; Boykov et al., 1999) and has been recently applied in natural language processing (Pang and Lee, 2004). The principal advantages of this formulation lie in its computational properties. Despite seeming intractable — the number of possible subsets to consider for selection is exponential in the number of database entities — the inference problem has an exact solution. Provided that the scores $ind_+(x)$, $ind_-(x)$, and $link_L(x,y)$ are positive, we can find a globally optimal label assignment in polynomial time by computing a minimal cut partition in an appropriately constructed graph (Greig et al., 1989).

In the following we first discuss how individual preference scores are estimated. Next, we describe how to induce links and estimate their scores.

3.2 Computing Individual Preference Scores

The individual preference scores are estimated by considering the values of entity attributes, recorded in the database. The type and number of the attributes are determined by the entity type. Therefore, we separately estimate individual preference scores for each entity type. For example, individual scores for entities of type `Passing` are computed based on six attributes : `PLAYER`, `CP/AT`, `YDS`, `AVG`, `TD`, `INT` (see Table 1).

Considerable latitude is available when selecting a classifier for delivering the individual preference scores. In our experiments we used the publicly available BoosTexter system (Schapire and Singer, 2000). BoosTexter implements a boosting algorithm that combines many simple, moderately accurate categorization rules into a single, highly accurate rule. For each example, it outputs a prediction along with a weight whose magnitude indicates the classifier's confidence in the prediction. We thus set the individual preference scores to the weights obtained from BoosTexter. The weights range from -1 to 1; we obtained non-negative numbers, simply by adding 1.

It is important to note that BoosTexter is a fairly effective classifier. When applied to text categorization (Schapire and Singer, 2000), it outperformed a number of alternative classification methods, including Naive Bayes, decision trees, and k-nearest neighbor.

3.3 Link Selection and Scoring

The success of collective classification depends on finding links between entities with similar label preferences. In our application — concept-to-text generation, it is natural to define entity links in terms of their database relatedness. Since the underlying database contains rich structural information, we can explore a wide range of relations between database entities.

The problem here is finding a set of links that capture important contextual dependencies among many possible combinations. Instead of manually specifying this set, we propose a corpus-driven method for discovering links automatically. Automatic link induction can greatly reduce human effort. Another advantage of the method is that it can potentially identify relations that might escape a human expert and yet, when explicitly modeled, aid in content selection.

We induce important links by adopting a generate-and-prune approach. We first automatically create a large pool of candidate links. Next, we select only links with aconsistent label distributions.

Construction of Candidate Links An important design decision is the type of links that we allow our algorithm to consider. Since our ultimate goal is the generation of a coherent text, we wish to focus on links that capture semantic connectivity between database entities. An obvious manifestation of semantic relatedness is attribute sharing. Therefore, we consider links across entities with one or more shared attributes. An additional constraint is implied by computational considerations: our optimization framework, based on minimal cuts in graphs, supports only pairwise links, so we restrict our attention to binary relations.

We generate a range of candidate link types using the following template: For every pair of entity types E_i and E_j, and for every attribute k that is associated with both of them, create a link of type $L_{i,j,k}$. A pair of entities $\langle a,b \rangle$ is linked by $L_{i,j,k}$, if a is of type E_i, b is of type E_j and they have the same value for the attribute k. For example, a link that associates statistics on `Passing` and `Rushing` performed by the same player is an instantiation of the above with E_i = Rushing, E_j = Passing, and k = Player.

In a similar fashion, we construct link types that connect together entities with two or three attributes in common. Multiple pairs of entries can be connected by the same link type.

If the database consists of n entity types, and the number of attribute types is bounded by m, then the number of link types constructed by this process does not exceed $O(n^2(m + \binom{m}{2} + \binom{m}{3})) \approx O(n^2 m^3)$. In practice, this bound is much lower, since only a few attributes are shared among entity types. Links can be efficiently computed using SQL's `SELECT` operator.

Link Filtering Only a small fraction of the automatically generated link types will capture meaningful contextual dependencies. To filter out spurious links, we turn to the labels of the entities participating in each link. Only link types in which entities have a similar distribution of label values are selected from the pool of candidates.

We measure similarity in label distribution using the χ^2 test. This test has been successfully applied to similar tasks, such as feature selection in text classification (Rogati and Yang, 2002), and can be easily extended to our application. Given a binary link, our null hypothesis H_0 is that the labels of entities related by L are independent. For each link, we compute the χ^2 score over a 2-by-2 table that stores joint label values of entity pairs, computed across all database entries present in the training set. For links with $\chi^2 > \tau$, the null hypothesis is rejected, and the link is considered a valid discourse constraint. The value of τ is set to 3.84, which corresponds to a 5% level of statistical significance.

Link Weights The score of a link type L is defined as follows:

$$link_L(x,y) = \begin{cases} \lambda_L & if(x,y) \text{ are linked by L} \\ 0 & \text{otherwise} \end{cases}$$

We estimate link weights λ_L using simulated annealing. The goal is to find weight values that minimize an objective function, defined as the error rate on the development set[5] (see Section 4 for details). The individual scores and the link structure of the entities in the development set are predicted automatically using the models trained on the training set. Starting from a random assignment of weight values, we compute the objective function and generate new weight values using Parks' (1990) method. The procedure stops when no sufficient progress is observed in subsequent iterations.

4 Evaluation Framework

We apply the collective classification method just presented to the task of automatically learning content selection rules from a database containing football-related information. In this section, we first present the sport domain we are working with, and

[5]Our objective function cannot be optimized analytically. We therefore resort to heuristic search methods such as simulated annealing.

Entity Type	Attr	Inst	%Aligned	Entity Type	Attr	Inst	%Aligned
Defense	8	14,077	0.00	*Passing*	5	1,185	59.90
Drive	10	11,111	0.00	*Team comparison*	4	14,539	0.00
Play-by-Play	8	83,704	3.03	*Punt-returns*	8	940	5.74
Fumbles	8	2,937	17.78	*Punting*	9	950	0.87
Game	6	469	0.00	*Receiving*	8	6,337	11.19
Interceptions	6	894	45.05	*Rushing*	8	3,631	9.17
Kicking	8	943	26.93	*Scoring-sum*	9	3,639	53.34
Kickoff-returns	8	1,560	5.24	*Team*	3	4	0.00
Officials	8	464	0.00				

Table 2: Entity types and their attributes in the NFL database; percentage of database entries that are aligned to summary sentences.

describe how we collected a corpus for evaluating collective content selection. Next, we explain how we automatically obtained annotated data for training and testing our model.

Data As mentioned previously our goal is to generate descriptions of football games. The sports domain has enjoyed popularity among natural language generation practitioners (Robin, 1994; Tanaka-Ishii et al., 1998). The appeal is partly due to the nature of the domain — it exhibits several fixed patterns in content organization and is therefore amenable to current generation approaches. At the same time, it is complex enough to present challenges at almost all stages of the generation process.

We compiled a corpus of descriptions of football games from the web. More specifically, we obtained game summaries from the official site of the American National Football League[6] (NFL). We collected summaries for the 2003 and 2004 seasons. These are typically written by Associated Press journalists. The corpus consists of 468 texts in total (436,580 words). The average summary length is 46.8 sentences.

The site not only contains a summary for each game, but also a wealth of statistics describing the performance of individual players and their teams. It includes a scoring summary and a play-by-play summary giving details of the most important events in the game together with temporal (i.e., time remaining) and positional (i.e., location in the field) information. In sum, for each game the site offers a rich repository of tabulated information which we translated into a relational database. An excerpt of the database is shown in Table 1. Table 2 displays the entity types contained in our NFL database and lists the number of attributes (Attr) and instantiations (Inst) per type. The database contains 73,400 entries in total.

[6]See `http://www.nfl.com/scores`.

Alignment Recall that our collective classification method is supervised. The training instances are database entries and the class labels indicate whether an instance should be selected for presentation or not. We could obtain this information via manual annotation performed by domain experts. Instead, we opted for a less costly, automatic solution that yields large quantities of training and testing data. To infer which database entries correspond to sentences in the verbalized game summaries, we used a simple anchor-based alignment technique. In our domain, numbers and proper names appear with high frequency, and they constitute reliable anchors for alignment. Similar to previous work (Duboue and McKeown, 2003; Sripada et al., 2001), we employ a simple matching procedure that considers anchor overlap between entity attributes and sentence tokens.

Overall, the alignment procedure produced 7,513 pairs. 7.1% of the database entries were verbalized in our corpus and 31.7% of the corpus sentences had a database entry. Table 2 presents the proportion of database entries which are verbalized in our corpus, broken down by entity type (see %Aligned).

To evaluate the accuracy of this procedure, we compared our output with a gold-standard alignment produced by a domain expert. After analyzing the data from five games, the expert produced 52 alignment pairs; 47 of these pairs were identified

	Majority Baseline			Standard Classifier			Collective Classifier		
	Prec	Rec	F-score	Prec	Rec	F-score	Prec	Rec	F-score
Mean	29.40	68.19	40.09	44.88	62.23	49.75	**52.71**	**76.50**	**60.15**
Min	3.57	28.57	6.45	12.50	8.33	13.33	12.50	27.27	19.05
Max	57.14	100.00	65.12	76.92	100.00	75.00	100.00	100.00	100.00
Std Dev	10.93	15.75	12.25	15.36	18.33	13.98	21.29	18.93	19.66

Table 3: Results on content selection (precision, recall and F-score are averages over individual game summaries); comparison between the majority baseline, standard and collective classification.

by the automatic alignment. In addition, three pairs produced by the program did not match the gold-standard alignment. Thus, the automatic method achieved 94.0% precision and 90.4% recall.

Data Annotation For training and testing purposes, we only considered entity types for which alignments were observed in our corpus (e.g., `Fumbles`, `Interceptions`; see Table 2). Types without alignments can be trivially regarded as inappropriate for selection in the generated text. We considered database entries for which we found verbalizations in the corpus as positive instances (i.e., they should be selected); accordingly, non-verbalized entries were considered negative instances (i.e., they should not be selected). The overall dataset contained 105,792 instances (corresponding to 468 game summaries). Of these, 15% (68 summaries) were reserved for testing. We held out 1,930 instances (10 summaries) from the training data for development purposes.

5 Results

Our results are summarized in Table 3. We compare the performance of the collective classifier against a standard classifier. This can be done in our framework, simply by setting the link scores to zero. We also report the performance of a majority baseline. The latter was obtained by defaulting to the majority class for each entity type in the training data. As can be seen from Table 2, only for two relations — `Passing` and `Scoring-sum` — the majority class predicts that the corresponding database instances should be selected for presentation.

Our results confirm that a content selection component can be automatically engineered for the football domain. The collective classifier achieves an F-score of 60.15%. This result compares favorably with Duboue and McKeown (2003) whose best model has an F-score of 51.00% on a simpler domain. Our method has high recall (we want to avoid missing out information that should be presented in the output) but tends to overgenerate as demonstrated by the relatively moderate precision in Table 3. Erroneous content selection decisions could be remedied by other components later in the generation process. Alternatively, the obtained content selection rules could be further refined or post-processed by a domain expert. Finally, better classification performance should be possible with more expressive feature sets. As we can see from the weak performance of the standard classifier, attribute values of database entries may not be sufficiently strong predictors. Considering additional features tailored to the NFL domain could further enhance performance. However, feature selection is not one of the main objectives of this work.

Our results empirically validate the importance of discourse constraints for content selection (Table 4 illustrates examples of constraints that the model discovered). We observe that adding contextual information leads to a 10.4% F-score increase over the standard classifier. We used a paired t test to examine whether the differences are statistically significant. The collective model significantly outperforms the standard model on both precision (t = 4.824, $p < 0.01$) and recall (t = 8.445, $p < 0.01$). It is also significantly better than the majority baseline, both in terms of recall (t = 3.181, $p < 0.01$) and precision (t = 8.604, $p < 0.01$). The standard classifier performs significantly better than the majority baseline on precision (t = 7.043, $p < 0.01$) but worse on recall (t =-2.274, $p < 0.05$).

6 Conclusions and Future Work

In this paper we have presented a novel, data-driven method for automating content selection. Central

$\{\langle a,b\rangle \mid a \in Sum \wedge b \in Sum \wedge a.Quarter = b.Quarter\}$
$\{\langle a,b\rangle \mid a \in Sum \wedge b \in Play \wedge Sum.Player_1 = Play.Player_1 \wedge Sum.Action = Play.Action\}$
$\{\langle a,b\rangle \mid a \in Fumbles \wedge b \in Interceptions \wedge Fumbles.Player = Interceptions.Player\}$

Table 4: Examples of automatically derived links.

to our approach is the use of a collective classification model that captures contextual dependencies between input items. We show that incorporation of discourse constraints yields substantial improvement over context-agnostic methods. Our approach is linguistically grounded, computationally efficient, and viable in practical applications.

In the future, we plan to explore how to integrate more refined discourse models in the content selection process. Currently, we consider a limited set of contextual dependencies based on attribute similarity. Ideally, we would like to express more complex relations between items. For instance, we may want to represent disjunctive constraints, such as "at least one of the defense players should be mentioned in the summary." Such dependencies can be efficiently handled in a collective classification framework by using approximate probabilistic inference (Taskar et al., 2002). Another promising approach is the combination of our automatically acquired cross-entity links with domain knowledge.

Needless to say, content selection is one of several components within a working generation system. An interesting question is how to integrate our component into a generation pipeline, using feedback from other components to guide collective content selection.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (Barzilay; CAREER grant IIS-0448168 and grant IIS-0415865) and EPSRC (Lapata; grant GR/T04540/01). We are grateful to Eli Barzilay for his help with data collection, and Luke Zettelmoyer who explained the many rules of American football to us. Thanks to Michael Collins, Amit Dubey, Noemie Elhadad, Dina Katabi, Frank Keller, Igor Malioutov, Smaranda Muresan, Martin Rinard, Kevin Simler and the anonymous reviewers for helpful comments and suggestions. Any opinions, fi ndings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the National Science Foundation or EPSRC.

References

J. Besag. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48:259–302.

Y. Boykov, O. Veksler, R. Zabih. 1999. Fast approximate energy minimization via graph cuts. In *ICCV*, 377–384.

P. A. Duboue, K. R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the EMNLP*, 121–128.

D. Greig, B. Porteous, A. Seheult. 1989. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2):271–279.

K. Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the ACL*, 145–150.

J. Kupiec, J. O. Pedersen, F. Chen. 1995. A trainable document summarizer. In *Proceedings of the SIGIR*, 68–73.

K. R. McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press.

B. Pang, L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, 271–278, Barcelona, Spain.

G. Parks. 1990. An intelligent stochastic optimization routine for nuclear fuel cycle design. *Nuclear Technology*, 89:233–246.

E. Reiter, R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

J. Robin. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. thesis, Columbia University.

M. Rogati, Y. Yang. 2002. High-performing feature selection for text classifi cation. In *Proceedings of the CIKM*, 659–661.

R. E. Schapire, Y. Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

S. G. Sripada, E. Reiter, J. Hunter, J. Yu. 2001. A two-stage model for content determination. In *Proceedings of the ACL-ENLG*, 3–10.

K. Tanaka-Ishii, K. Hasida, I. Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the ACL/COLING*, 1282–1288.

B. Taskar, P. Abbeel, D. Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the UAI*, 485–495.

Extracting Product Features and Opinions from Reviews

Ana-Maria Popescu and Oren Etzioni
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
{amp, etzioni}@cs.washington.edu

Abstract

Consumers are often forced to wade through many on-line reviews in order to make an informed product choice. This paper introduces OPINE, an unsupervised information-extraction system which mines reviews in order to build a model of important product features, their evaluation by reviewers, and their relative quality across products.

Compared to previous work, OPINE achieves 22% higher precision (with only 3% lower recall) on the feature extraction task. OPINE's novel use of *relaxation labeling* for finding the semantic orientation of words in context leads to strong performance on the tasks of finding opinion phrases and their polarity.

1 Introduction

The Web contains a wealth of opinions about products, politicians, and more, which are expressed in newsgroup posts, review sites, and elsewhere. As a result, the problem of "opinion mining" has seen increasing attention over the last three years from (Turney, 2002; Hu and Liu, 2004) and many others. This paper focuses on product reviews, though our methods apply to a broader range of opinions.

Product reviews on Web sites such as `amazon.com` and elsewhere often associate meta-data with each review indicating how positive (or negative) it is using a 5-star scale, and also rank products by how they fare in the reviews at the site. However, the reader's taste may differ from the reviewers'. For example, the reader may feel strongly about the quality of the gym in a hotel, whereas many reviewers may focus on other aspects of the hotel, such as the decor or the location. Thus, the reader is forced to wade through a large number of reviews looking for information about particular features of interest.

We decompose the problem of review mining into the following main subtasks:

I. Identify product features.
II. Identify opinions regarding product features.
III. Determine the polarity of opinions.
IV. Rank opinions based on their strength.

This paper introduces OPINE, an unsupervised information extraction system that embodies a solution to each of the above subtasks. OPINE is built on top of the KnowItAll Web information-extraction system (Etzioni et al., 2005) as detailed in Section 3.

Given a particular product and a corresponding set of reviews, OPINE solves the opinion mining tasks outlined above and outputs a set of *product features*, each accompanied by a list of *associated opinions* which are ranked based on strength (*e.g.*, "abominable" is stronger than "bad). This output information can then be used to generate various types of opinion summaries.

This paper focuses on the first 3 review mining subtasks and our contributions are as follows:

1. We introduce OPINE, a review-mining system whose novel components include the use of *relaxation labeling* to find the semantic orientation of words in the context of given product features and sentences.

2. We compare OPINE with the most relevant previous review-mining system (Hu and Liu, 2004) and find that OPINE's precision on the *feature extraction* task is 22% better though its recall is 3% lower on Hu's data sets. We show that 1/3 of this increase in precision comes from using OPINE's *feature assessment* mechanism on review data while the rest is due to Web PMI statistics.

3. While many other systems have used extracted opinion phrases in order to determine the polarity of sentences or documents, OPINE is the first to report its precision and recall on the tasks of *opinion phrase extraction* and *opinion phrase polarity determination* in the context of known product features and sentences. On the first task, OPINE has a precision of 79% and a recall of 76%. On the second task, OPINE has a precision of 86% and a recall of 89%.

```
Input: product class C, reviews R.
Output: set of [feature, ranked opinion list] tuples
 R' ← parseReviews(R);
 E ← findExplicitFeatures(R', C);
 O ← findOpinions(R', E);
 CO ← clusterOpinions(O);
 I ← findImplicitFeatures(CO, E);
 RO ← rankOpinions(CO);
 {($f$, $o_i$, ...$o_j$)...}←outputTuples(RO, I ∪ E);
```

Figure 1: **OPINE Overview.**

The remainder of this paper is organized as follows: Section 2 introduces the basic terminology, Section 3 gives an overview of OPINE, describes and evaluates its main components, Section 4 describes related work and Section 5 presents our conclusion.

2 Terminology

A *product class* (*e.g.*, Scanner) is a set of *products* (*e.g.*, Epson1200). OPINE extracts the following types of *product features*: *properties*, *parts*, *features of product parts*, *related concepts*, *parts* and *properties of related concepts* (see Table 1 for examples of such features in the Scanner domains). *Related concepts* are concepts relevant to the customers' experience with the main product (*e.g.*, the company that manufactures a scanner). The relationships between the main product and related concepts are typically expressed as verbs (*e.g.*, "Epson *manufactures* scanners") or prepositions ("scanners *from* Epson"). Features can be *explicit* ("good `scan quality`") or *implicit* ("good scans" implies good `ScanQuality`).

OPINE also extracts *opinion phrases*, which are adjective, noun, verb or adverb phrases representing customer opinions. Opinions can be *positive* or *negative* and vary in *strength* (*e.g.*, "fantastic" is stronger than "good").

3 OPINE Overview

This section gives an overview of OPINE (see Figure 1) and describes its components and their experimental evaluation.

Goal Given product class C with instances I and reviews R, OPINE's goal is to find a set of (feature, opinions) tuples $\{(f, o_i, ...o_j)\}$ s.t. $f \in F$ and $o_i, ...o_j \in O$, where:

a) F is the set of product class features in R.

b) O is the set of opinion phrases in R.

c) f is a feature of a particular product instance.

d) o is an opinion about f in a particular sentence.

d) the opinions associated with each feature f are ranked based on their strength.

Solution The steps of our solution are outlined in Figure 1 above. OPINE parses the reviews using MINIPAR (Lin, 1998) and applies a simple pronoun-resolution module to parsed review data. OPINE then uses the data to find *explicit* product features (E). OPINE's *Feature Assessor* and its use of Web PMI statistics are vital for the extraction of high-quality features (see 3.2). OPINE then identifies *opinion phrases* associated with features in E and finds their polarity. OPINE's novel use of relaxation-labeling techniques for determining the semantic orientation of potential opinion words in the context of given features and sentences leads to high precision and recall on the tasks of *opinion phrase extraction* and *opinion phrase polarity extraction* (see 3.3).

In this paper, we only focus on the extraction of explicit features, identifying corresponding customer opinions about these features and determining their polarity. We omit the descriptions of the opinion clustering, implicit feature generation and opinion ranking algorithms.

3.0.1 The KnowItAll System.

OPINE is built on top of KnowItAll, a Web-based, domain-independent information extraction system (Etzioni et al., 2005). Given a set of relations of interest, KnowItAll instantiates relation-specific generic extraction patterns into extraction rules which find candidate facts. KnowItAll's Assessor then assigns a probability to each candidate. The Assessor uses a form of *Point-wise Mutual Information* (PMI) between phrases that is estimated from Web search engine hit counts (Turney, 2001). It computes the PMI between each fact and *automatically generated discriminator phrases* (*e.g.*, "is a scanner" for the `isA()` relationship in the context of the `Scanner` class). Given fact f and discriminator d, the computed PMI score is:

$$\text{PMI}(f, d) = \frac{\text{Hits}(d + f)}{\text{Hits}(d) * \text{Hits}(f)}$$

The PMI scores are converted to binary features for a Naive Bayes Classifier, which outputs a probability associated with each fact (Etzioni et al., 2005).

3.1 Finding Explicit Features

OPINE extracts *explicit* features for the given product class from parsed review data. First, the system recursively identifies both the *parts* and the *properties* of the given product class and their parts and properties, in turn, continuing until no candidates are found. Then, the system finds *related concepts* as described in (Popescu et al., 2004) and extracts their parts and properties. Table 1 shows that each feature type contributes to the set of final features (averaged over 7 product classes).

Explicit Features	**Examples**	**% Total**
Properties	ScannerSize	7%
Parts	ScannerCover	52%
Features of Parts	BatteryLife	24%
Related Concepts	ScannerImage	9%
Related Concepts' Features	ScannerImageSize	8%

Table 1: **Explicit Feature Information**

In order to find parts and properties, OPINE first extracts the noun phrases from reviews and retains those with frequency greater than an experimentally set threshold. OPINE's *Feature Assessor*, which is an instantiation of KnowItAll's Assessor, evaluates each noun phrase by computing the PMI scores between the phrase and *meronymy discriminators* associated with the product class (*e.g.*, "of scanner", "scanner has", "scanner comes with", etc. for the `Scanner` class). OPINE distinguishes parts from properties using WordNet's IS-A hierarchy (which enumerates different kinds of properties) and morphological cues (*e.g.*, "-iness", "-ity" suffixes).

3.2 Experiments: Explicit Feature Extraction

In our experiments we use sets of reviews for 7 product classes (1621 total reviews) which include the publicly available data sets for 5 product classes from (Hu and Liu, 2004). Hu's system is the review mining system most relevant to our work. It uses association rule mining to extract *frequent* review noun phrases as features. Frequent features are used to find *potential opinion* words (only adjectives) and the system uses WordNet synonyms/antonyms in conjunction with a set of seed words in order to find actual *opinion* words. Finally, opinion words are used to extract associated *infrequent* features. The system only extracts *explicit* features.

On the 5 datasets in (Hu and Liu, 2004), OPINE's precision is 22% higher than Hu's at the cost of a 3% recall drop. There are two important differences between OPINE and Hu's system: a) OPINE's Feature Assessor uses PMI assessment to evaluate each candidate feature and b) OPINE incorporates Web PMI statistics in addition to review data in its assessment. In the following, we quantify the performance gains from a) and b).

a) In order to quantify the benefits of OPINE's Feature Assessor, we use it to evaluate the features extracted by Hu's algorithm on review data (**Hu+A/R**). The Feature Assessor improves Hu's precision by 6%.

b) In order to evaluate the impact of using Web PMI statistics, we assess OPINE's features first on reviews (**OP/R**) and then on reviews in conjunction with the Web (the corresponding methods are **Hu+A/R+W** and **OPINE**). Web PMI statistics increase precision by an average of 14.5%.

Overall, 1/3 of **OPINE**'s precision increase over Hu's system comes from using PMI assessment on reviews and the other 2/3 from the use of the Web PMI statistics.

In order to show that OPINE's performance is robust across multiple product classes, we used two sets of reviews downloaded from `tripadvisor.com` for Hotels and `amazon.com` for Scanners. Two annotators labeled a set of unique 450 OPINE extractions as *correct* or *incorrect*. The inter-annotator agreement was 86%. The extractions on which the annotators agreed were used to compute OPINE's precision, which was 89%. Furthermore, the annotators extracted explicit features from 800 review sentences (400 for each domain). The inter-annotator agreement was 82%. OPINE's recall on the set of 179 features on which both annotators agreed was 73%.

Data	**Explicit Feature Extraction: Precision**				
	Hu	**Hu+A/R**	**Hu+A/R+W**	**OP/R**	**OPINE**
D_1	*0.75*	+0.05	+0.17	+0.07	**+0.19**
D_2	*0.71*	+0.03	+0.19	+0.08	**+0.22**
D_3	*0.72*	+0.03	+0.25	+0.09	**+0.23**
D_4	*0.69*	+0.06	+0.22	+0.08	**+0.25**
D_5	*0.74*	+0.08	+0.19	+0.04	**+0.21**
Avg	*0.72*	+0.06	+ 0.20	+0.07	**+0.22**

Table 2: **Precision Comparison on the Explicit Feature-Extraction Task.** OPINE's precision is 22% better than Hu's precision; Web PMI statistics are responsible for 2/3 of the precision increase. All results are reported with respect to Hu's.

Data	**Explicit Feature Extraction: Recall**				
	Hu	**Hu+A/R**	**Hu+A/R+W**	**OP/R**	**OPINE**
D_1	*0.82*	-0.16	-0.08	-0.14	**-0.02**
D_2	*0.79*	-0.17	-0.09	-0.13	**-0.06**
D_3	*0.76*	-0.12	-0.08	-0.15	**-0.03**
D_4	*0.82*	-0.19	-0.04	-0.17	**-0.03**
D_5	*0.80*	-0.16	-0.06	-0.12	**-0.02**
Avg	*0.80*	-0.16	-0.07	-0.14	**-0.03**

Table 3: **Recall Comparison on the Explicit Feature-Extraction Task.** OPINE's recall is 3% lower than the recall of Hu's original system (precision level = 0.8). All results are reported with respect to Hu's.

3.3 Finding Opinion Phrases and Their Polarity

This subsection describes how OPINE extracts potential opinion phrases, distinguishes between opinions and non-opinions, and finds the *polarity* of each opinion in the context of its associated feature in a particular review sentence.

3.3.1 Extracting Potential Opinion Phrases

OPINE uses explicit features to identify potential opinion phrases. Our intuition is that an opinion phrase associated with a product feature will occur in its vicinity. This idea is similar to that of (Kim and Hovy, 2004) and (Hu and Liu, 2004), but instead of using a window of size k or the output of a noun phrase chunker, OPINE takes advantage of the syntactic dependencies computed by the MINIPAR parser. Our intuition is embodied by 10 *extraction rules*, some of which are shown in Table 4. If an explicit feature is found in a sentence, OPINE applies the extraction rules in order to find the heads of potential opinion phrases. Each head word together with its modifiers is returned as a potential opinion phrase[1].

Extraction Rules	**Examples**
if $\exists(M, NP = f) \rightarrow po = M$	(expensive) `scanner`
if $\exists(S = f, P, O) \rightarrow po = O$	`lamp` has (problems)
if $\exists(S, P, O = f) \rightarrow po = P$	I (hate) this `scanner`
if $\exists(S = f, P, O) \rightarrow po = P$	`program` (crashed)

Table 4: **Examples of Domain-independent Rules for the Extraction of Potential Opinion Phrases.** Notation: po=potential opinion, M=modifier, NP=noun phrase, S=subject, P=predicate, O=object. Extracted phrases are enclosed in parentheses. Features are indicated by the typewriter font. The equality conditions on the left-hand side use po's head.

Rule Templates	**Rules**
$dep(w, w')$	$m(w, w')$
$\exists v$ s.t. $dep(w, v), dep(v, w')$	$\exists v$ s.t. $m(w, v), o(v, w')$
$\exists v$ s.t. $dep(w, v), dep(w', v)$	$\exists v$ s.t. $m(w, v), o(w', v)$

Table 5: **Dependency Rule Templates For Finding Words w, w' with Related SO Labels** . OPINE instantiates these templates in order to obtain extraction rules. Notation: dep=dependent, m=modifier, o=object, v,w,w'=words.

OPINE examines the potential opinion phrases in order to identify the actual opinions. First, the system finds the semantic orientation for the lexical head of each potential opinion phrase. Every phrase whose head word has a *positive* or *negative* semantic orientation is then retained as an *opinion phrase*. In the following, we describe how OPINE finds the semantic orientation of words.

3.3.2 Word Semantic Orientation

OPINE finds the semantic orientation of a word w in the context of an associated feature f and sentence s. We restate this task as follows:

Task Given a set of *semantic orientation (SO) labels* ($\{positive, negative, neutral\}$), a set of reviews and a set of tuples (w, f, s), where w is a potential opinion word associated with feature f in sentence s, assign a SO label to each tuple (w, f, s).

For example, the tuple (*sluggish*, *driver*, "I am not happy with this sluggish driver") would be assigned a *negative* SO label.

Note: We use "word" to refer to a potential opinion word w and "feature" to refer to the word or phrase which represents the explicit feature f.

Solution OPINE uses the 3-step approach below:

1. Given the set of reviews, OPINE finds a SO label for each word w.

2. Given the set of reviews and the set of SO labels for words w, OPINE finds a SO label for each (w, f) pair.

[1]The (S,P,O) tuples in Table 4 are automatically generated from MINIPAR's output.

3. Given the set of SO labels for (w, f) pairs, OPINE finds a SO label for each (w, f, s) input tuple.

Each of these subtasks is cast as an *unsupervised collective classification* problem and solved using the same mechanism. In each case, OPINE is given a set of *objects* (words, pairs or tuples) and a set of *labels* (SO labels); OPINE then searches for a *global* assignment of labels to objects. In each case, OPINE makes use of *local constraints* on label assignments (*e.g.*, conjunctions and disjunctions constraining the assignment of SO labels to words (Hatzivassiloglou and McKeown, 1997)).

A key insight in OPINE is that the problem of searching for a *global* SO label assignment to words, pairs or tuples while trying to satisfy as many *local* constraints on assignments as possible is analogous to labeling problems in computer vision (*e.g.*, model-based matching). OPINE uses a well-known computer vision technique, *relaxation labeling* (Hummel and Zucker, 1983), in order to solve the three subtasks described above.

3.3.3 Relaxation Labeling Overview

Relaxation labeling is an unsupervised classification technique which takes as input:

a) a set of *objects* (*e.g.*, words)
b) a set of *labels* (*e.g.*, SO labels)
c) initial probabilities for each object's possible labels
d) the definition of an object o's *neighborhood* (a set of other objects which influence the choice of o's label)
e) the definition of *neighborhood features*
f) the definition of a *support function* for an object label

The influence of an object o's neighborhood on its label L is quantified using the *support function*. The support function computes the probability of the label L being assigned to o as a function of o's *neighborhood features*. Examples of features include the fact that a certain *local constraint* is satisfied (*e.g.*, the word $nice$ participates in the conjunction *and* together with some other word whose SO label is estimated to be *positive*).

Relaxation labeling is an iterative procedure whose output is an assignment of labels to objects. At each iteration, the algorithm uses an *update equation* to reestimate the probability of an object label based on its previous probability estimate and the features of its neighborhood. The algorithm stops when the global label assignment stays constant over multiple consecutive iterations.

We employ relaxation labeling for the following reasons: a) it has been extensively used in computer-vision with good results b) its formalism allows for many types of constraints on label assignments to be used simultaneously. As mentioned before, constraints are integrated into the algorithm as neighborhood features which influence the assignment of a particular label to a particular object.

OPINE uses the following sources of constraints:

a) *conjunctions* and *disjunctions* in the review text

b) manually-supplied *syntactic dependency rule templates* (see Table 5). The templates are automatically instantiated by our system with different dependency relationships (premodifier, postmodifier, subject, etc.) in order to obtain syntactic dependency rules which find words with related SO labels.

c) automatically derived *morphological relationships* (*e.g.*, "wonderful" and "wonderfully" are likely to have similar SO labels).

d) WordNet-supplied *synonymy, antonymy, IS-A* and *morphological* relationships between words. For example, *clean* and *neat* are synonyms and so they are likely to have similar SO labels.

Each of the SO label assignment subtasks previously identified is solved using a relaxation labeling step. In the following, we describe in detail how relaxation labeling is used to find SO labels for words in the given review sets.

3.3.4 Finding SO Labels for Words

For many words, a word sense or set of senses is used throughout the review corpus with a consistently positive, negative or neutral connotation (*e.g.*, "great", "awful", etc.). Thus, in many cases, a word w's SO label in the context of a feature f and sentence s will be the same as its SO label in the context of other features and sentences. In the following, we describe how OPINE's relaxation labeling mechanism is used to find a word's dominant SO label in a set of reviews.

For this task, a word's *neighborhood* is defined as the set of words connected to it through conjunctions, disjunctions and all other relationships previously introduced as sources of constraints.

RL uses an *update equation* to re-estimate the probability of a word label based on its previous probability estimate and the features of its neighborhood (see **Neighborhood Features**). At iteration m, let $q(w, L)_{(m)}$ denote the support function for label L of w and let $P(l(w) = L)_{(m)}$ denote the probability that L is the label of w. $P(l(w) = L)_{(m+1)}$ is computed as follows:

RL Update Equation (Rangarajan, 2000)

$$P(l(w) = L)_{(m+1)} = \frac{P(l(w) = L)_{(m)}(1 + \alpha q(w, L)_{(m)})}{\sum_{L'} P(l(w) = L')_{(m)}(1 + \alpha q(w, L')_{(m)})}$$

where $L' \in \{pos, neg, neutral\}$ and $\alpha > 0$ is an experimentally set constant keeping the numerator and probabilities positive. RL's output is an assignment of dominant SO labels to words.

In the following, we describe in detail the initialization step, the derivation of the support function formula and the use of neighborhood features.

RL Initialization Step OPINE uses a version of Turney's PMI-based approach (Turney, 2003) in order to derive the initial probability estimates ($P(l(w) = L)_{(0)}$) for a subset S of the words. OPINE computes a *SO score so(w)* for each w in S as the difference between the PMI of w with positive keywords (*e.g.*, "excellent") and the PMI of w with negative keywords (*e.g.*, "awful"). When $so(w)$ is small, or w rarely co-occurs with the keywords, w is classified as *neutral*. If $so(w) > 0$, then w is *positive*, otherwise w is negative. OPINE then uses the labeled S set in order to compute prior probabilities $P(l(w) = L)$, $L \in \{pos, neg, neutral\}$ by computing the ratio between the number of words in S labeled L and $|S|$. Such probabilities are used as initial probability estimates associated with the labels of the remaining words.

Support Function The support function computes the probability of each label for word w based on the labels of objects in w's neighborhood N.

Let $A_k = \{(w_j, L_j)|w_j \in N\}$, $0 < k \leq 3^{|N|}$ represent one of the potential assignments of labels to the words in N. Let $P(A_k)_{(m)}$ denote the probability of this particular assignment at iteration m. The *support* for label L of word w at iteration m is :

$$q(w, L)_{(m)} = \sum_{k=1}^{3^{|N|}} P(l(w) = L|A_k)_{(m)} * P(A_k)_{(m)}$$

We assume that the labels of w's neighbors are independent of each other and so the formula becomes:

$$q(w, L)_{(m)} = \sum_{k=1}^{3^{|N|}} P(l(w) = L|A_k)_{(m)} * \prod_{j=1}^{|N|} P(l(w_j) = L_j)_{(m)}$$

Every $P(l(w_j) = L_j)_{(m)}$ term is the estimate for the probability that $l(w_j) = L_j$ (which was computed at iteration m using the RL update equation).

The $P(l(w) = L|A_k)_{(m)}$ term quantifies the influence of a particular label assignment to w's neighborhood over w's label. In the following, we describe how we estimate this term.

Neighborhood Features

Each type of word relationship which constrains the assignment of SO labels to words (synonymy, antonymy, etc.) is mapped by OPINE to a neighborhood feature. This mapping allows OPINE to use simultaneously use multiple independent sources of constraints on the label of a particular word. In the following, we formalize this mapping.

Let T denote the type of a word relationship in R (synonym, antonym, etc.) and let $A_{k,T}$ represent the labels assigned by A_k to neighbors of a word w which are connected to w through a relationship of type T . We have $A_k = \bigcup_T A_{k,T}$ and

$$P(l(w) = L|A_k)_{(m)} = P(l(w) = L|\bigcup_T A_{k,T})_{(m)}$$

For each relationship type T, OPINE defines a *neighborhood feature* $f_T(w, L, A_{k,T})$ which computes $P(l(w) = L|A_{k,T})$, the probability that w's label is L given $A_{k,T}$ (see below). $P(l(w) = L|\bigcup_T A_{k,T})_{(m)}$ is estimated combining the information from various features about w's label using the sigmoid function $\sigma()$:

$$P(l(w) = L|A_k)_{(m)} = \sigma(\sum_{i=1}^{j} f_i(w, L, A_{k,i})_{(m)} * c_i)$$

where $c_0, ...c_j$ are weights whose sum is 1 and which reflect OPINE 's confidence in each type of feature.

Given word w, label L, relationship type T and neighborhood label assignment A_k, let N_T represent the subset of w's neighbors connected to w through a type T relationship. The feature f_T computes the probability that w's label is L given the labels assigned by A_k to words in N_T. Using Bayes's Law and assuming that these labels are independent given $l(w)$, we have the following formula for f_T at iteration m:

$$f_T(w, L, A_{k,T})_{(m)} = P(l(w) = L)_{(m)} * \prod_{j=1}^{|N_T|} P(L_j|l(w) = L)$$

$P(L_j|l(w) = L)$ is the probability that word w_j has label L_j if w_j and w are linked by a relationship of type T and w has label L. We make the simplifying assumption that this probability is constant and depends only of T, L and L', not of the particular words w_j and w. For each tuple (T, L, L_j), $L, L_j \in \{pos, neg, neutral\}$, OPINE builds a probability table using a small set of bootstrapped positive, negative and neutral words.

3.3.5 Finding (Word, Feature) SO Labels

This subtask is motivated by the existence of frequent words which change their SO label based on associated features, but whose SO labels in the context of the respective features are consistent throughout the reviews (*e.g.,* in the Hotel domain, "hot water" has a consistently positive connotation, whereas "hot room" has a negative one).

In order to solve this task, OPINE first assigns each (w, f) pair an initial SO label which is w's SO label. The system then executes a relaxation labeling step during which syntactic relationships between words and, respectively, between features, are used to update the default SO labels whenever necessary. For example, *(hot, room)* appears in the proximity of *(broken, fan)*. If "room"and "fan" are conjoined by *and*, this suggests that "hot" and "broken" have similar SO labels in the context of their respective features. If "broken" has a strongly negative semantic orientation, this fact contributes to OPINE's belief that "hot" may also be negative in this context. Since *(hot, room)* occurs in the vicinity of other such phrases (*e.g., stifling kitchen*), "hot" acquires a negative SO label in the context of "room".

3.3.6 Finding (Word, Feature, Sentence) SO Labels

This subtask is motivated by the existence of (w,f) pairs (*e.g., (big, room)*) for which w's orientation changes based on the sentence in which the pair appears (*e.g.,* " I hated the big, drafty room because I ended up freezing." vs. "We had a big, luxurious room".)

In order to solve this subtask, OPINE first assigns each (w, f, s) tuple an initial label which is simply the SO label for the (w, f) pair. The system then uses syntactic relationships between words and, respectively, features in order to update the SO labels when necessary. For example, in the sentence "I hated the big, drafty room because I ended up freezing.", "big" and "hate" satisfy condition 2 in Table 5 and therefore OPINE expects them to have similar SO labels. Since "hate" has a strong negative connotation, "big" acquires a negative SO label in this context.

In order to correctly update SO labels in this last step, OPINE takes into consideration the presence of *negation modifiers*. For example, in the sentence "I don't like a large scanner either", OPINE first replaces the *positive* (w, f) pair *(like, scanner)* with the *negative* labeled pair *(not like, scanner)* and then infers that "large" is likely to have a negative SO label in this context.

3.3.7 Identifying Opinion Phrases

After OPINE has computed the most likely SO labels for the head words of each potential opinion phrase in the context of given features and sentences, OPINE can extract opinion phrases and establish their polarity. Phrases whose head words have been assigned *positive* or *negative* labels are retained as *opinion phrases*. Furthermore, the polarity of an opinion phrase o in the context of a feature f and sentence s is given by the SO label assigned to the tuple $(head(o), f, s)$ (3.3.6 shows how OPINE takes into account negation modifiers).

3.4 Experiments

In this section we evaluate OPINE's performance on the following tasks: finding SO labels of words in the context of known features and sentences (*SO label extraction*); distinguishing between opinion and non-opinion phrases in the context of known features and sentences (*opinion phrase extraction*); finding the correct polarity of extracted opinion phrases in the context of known features and sentences (*opinion phrase polarity extraction*).

While other systems, such as (Hu and Liu, 2004; Turney, 2002), have addressed these tasks to some degree, OPINE is the first to report results. We first ran OPINE on 13841 sentences and 538 previously extracted features. OPINE searched for a SO label assignment for 1756 different words in the context of the given features and sentences. We compared OPINE against two baseline methods, **PMI++** and **Hu++**.

PMI++ is an extended version of (Turney, 2002)'s method for finding the SO label of a phrase (as an attempt to deal with context-sensitive words). For a given (word, feature, sentence) tuple, **PMI++** ignores the sentence, generates a phrase based on the word and the feature (*e.g., (clean, room)*: "clean room") and finds its SO label using PMI statistics. If unsure of the label, **PMI++** tries to find the orientation of the potential opinion word instead. The search engine queries use domain-specific keywords (*e.g.,* "scanner"), which are dropped if they lead to low counts.

Hu++ is a WordNet-based method for finding a word's context-independent semantic orientation. It extends Hu's adjective labeling method in a number of ways in order to handle nouns, verbs and adverbs in addition to adjectives and in order to improve coverage. Hu's method starts with two sets of positive and negative words and iteratively grows each one by including synonyms and antonyms from WordNet. The final sets are used to predict the orientation of an incoming word.

Type	PMI++		Hu++		OPINE	
	P	R	P	R	P	R
adj	0.73	0.91	+0.02	-0.17	**+0.07**	**-0.03**
nn	0.63	0.92	+0.04	-0.24	**+0.11**	**-0.08**
vb	0.71	0.88	+0.03	-0.12	**+0.01**	**-0.01**
adv	0.82	0.92	+0.02	-0.01	**+0.06**	**+0.01**
Avg	0.72	0.91	+0.03	-0.14	**+0.06**	**-0.03**

Table 6: **Finding SO Labels of Potential Opinion Words in the Context of Given Product Features and Sentences.** OPINE's precision is higher than that of **PMI++** and **Hu++**. All results are reported with respect to **PMI++** . Notation: adj=adjectives, nn=nouns, vb=verbs, adv=adverbs

3.4.1 Experiments: SO Labels

On the task of *finding SO labels for words in the context of given features and review sentences*, OPINE obtains higher precision than both baseline methods at a small loss in recall with respect to **PMI++**. As described below, this result is due in large part to OPINE's ability to handle context-sensitive opinion words.

We randomly selected 200 (word, feature, sentence) tuples for each word type (adjective, adverb, etc.) and obtained a test set containing 800 tuples. Two annotators assigned positive, negative and neutral labels to each tuple (the inter-annotator agreement was 78%). We retained the tuples on which the annotators agreed as the gold standard. We ran **PMI++** and **Hu++** on the test data and compared the results against OPINE's results on the same data.

In order to quantify the benefits of each of the three steps of our method for finding SO labels, we also compared OPINE with a version which only finds SO labels for words and a version which finds SO labels for words in the context of given features, but doesn't take into account given sentences. We have learned from this comparison that OPINE's precision gain over **PMI++** and **Hu++** is mostly due to to its ability to handle context-sensitive words in a large number of cases.

Although **Hu++** does not handle context-sensitive SO label assignment, its average precision was reasonable (75%) and better than that of **PMI++**. Finding a word's SO label is good enough in the case of strongly positive or negative opinion words, which account for the majority of opinion instances. The method's loss in recall is due to not recognizing words absent from WordNet (*e.g.*, "depth-adjustable") or not having enough information to classify some words in WordNet.

PMI++ typically does well in the presence of strongly positive or strongly negative words. Its high recall is correlated with decreased precision, but overall this simple approach does well. **PMI++**'s main shortcoming is misclassifying terms such as "basic" or "visible" which change orientation based on context.

3.4.2 Experiments: Opinion Phrases

In order to evaluate OPINE on the tasks of *opinion phrase extraction* and *opinion phrase polarity extraction in the context of known features and sentences*, we used a set of 550 sentences containing previously extracted features. The sentences were annotated with the opinion phrases corresponding to the known features and with the opinion polarity. We compared OPINE with **PMI++** and **Hu++** on the tasks of interest. We found that OPINE had the highest precision on both tasks at a small loss in recall with respect to **PMI++**. OPINE's ability to identify a word's SO label in the context of a given feature and sentence allows the system to correctly extract opinions expressed by words such as "big" or "small", whose semantic orientation varies based on context.

Measure	PMI++	Hu++	OPINE
OP Extraction: Precision	0.71	+0.06	**+0.08**
OP Extraction: Recall	0.78	-0.08	**-0.02**
OP Polarity: Precision	0.80	-0.04	**+0.06**
OP Polarity: Recall	0.93	+0.07	**-0.04**

Table 7: **Extracting Opinion Phrases and Opinion Phrase Polarity Corresponding to Known Features and Sentences.** OPINE's precision is higher than that of **PMI++** and of **Hu++**. All results are reported with respect to **PMI++**.

4 Related Work

The key components of OPINE described in this paper are the PMI feature assessment which leads to high-precision feature extraction and the use of relaxation-labeling in order to find the semantic orientation of potential opinion words. The review-mining work most relevant to our research is that of (Hu and Liu, 2004) and (Kobayashi et al., 2004). Both identify product features from reviews, but OPINE significantly improves on both. (Hu and Liu, 2004) doesn't assess candidate features, so its precision is lower than OPINE's. (Kobayashi et al., 2004) employs an iterative semi-automatic approach which requires human input at every iteration. Neither model explicitly addresses *composite* (feature of feature) or *implicit* features. Other systems (Morinaga et al., 2002; Kushal et al., 2003) also look at Web product reviews but they do not extract opinions about particular product features. OPINE's use of meronymy lexico-syntactic patterns is similar to that of many others, from (Berland and Charniak, 1999) to (Almuhareb and Poesio, 2004).

Recognizing the subjective character and polarity of words, phrases or sentences has been addressed by many authors, including (Turney, 2003; Riloff et al., 2003; Wiebe, 2000; Hatzivassiloglou and McKeown, 1997). Most recently, (Takamura et al., 2005) reports on the use of spin models to infer the semantic orientation of words. The paper's global optimization approach and use of multiple sources of constraints on a word's semantic orientation is similar to ours, but the mechanism differs and they currently omit the use of syntactic information. Subjective phrases are used by (Turney, 2002; Pang and Vaithyanathan, 2002; Kushal et al., 2003; Kim and Hovy, 2004) and others in order to classify reviews or sentences as positive or negative. So far, OPINE's focus has been on extracting and analyzing opinion phrases corresponding to specific features in specific sentences, rather than on determining sentence or review polarity.

5 Conclusion

OPINE is an unsupervised information extraction system which extracts fine-grained features, and associated opinions, from reviews. OPINE's use of the Web as a corpus helps identify product features with improved precision compared with previous work. OPINE uses a novel relaxation-labeling technique to determine the semantic orientation of potential opinion words in the context of the extracted product features and specific review sentences; this technique allows the system to identify customer opinions and their polarity with high precision and recall.

6 Acknowledgments

We would like to thank the KnowItAll project and the anonymous reviewers for their comments. Michael Gamon, Costas Boulis and Adam Carlson have also provided valuable feedback. We thank Minquing Hu and Bing Liu for providing their data sets and for their comments. Finally, we are grateful to Bernadette Minton and Fetch Technologies for their help in collecting additional reviews. This research was supported in part by NSF grant IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-02-1-0324 as well as gifts from Google and the Turing Center.

References

A. Almuhareb and M. Poesio. 2004. Attribute-based and value-based clustering: An evaluation. In *EMNLP*, pages 158–165.

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *ACL*, pages 57–64.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

V. Hatzivassiloglou and K. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL/EACL*, pages 174–181.

M. Hu and B. Liu. 2004. Mining and Summarizing Customer Reviews. In *KDD*, pages 168–177, Seattle, WA.

R.A. Hummel and S.W. Zucker. 1983. On the foundations of relaxation labeling processes. In *PAMI*, pages 267–287.

S. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *COLING*.

N. Kobayashi, K. Inui, K. Tateishi, and T. Fukushima. 2004. Collecting Evaluative Expressions for Opinion Extraction. In *IJCNLP*, pages 596–605.

D. Kushal, S. Lawrence, and D. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW*.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on Evaluation of Parsing Systems at ICLRE*.

S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. 2002. Mining product reputations on the web. In *KDD*.

Lee L. Pang, B and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

A. Popescu, A. Yates, and O. Etzioni. 2004. Class extraction from the World Wide Web. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*, pages 68–73.

A. Rangarajan. 2000. Self annealing and self annihilation: unifying deterministic annealing and relaxation labeling. In *Pattern Recognition, 33:635-649*.

E. Riloff, J. Wiebe, and T. Wilson. 2003. Learning Subjective Nouns Using Extraction Pattern Bootstrapping. In *CoNLL*, pages 25–32s.

H. Takamura, T. Inui, and M. Okumura. 2005. Extracting Semantic Orientations of Words using Spin Model. In *ACL*, pages 133–141.

P. D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Procs. of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, Freiburg, Germany.

P. D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Procs. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424.

P. Turney. 2003. Inference of Semantic Orientation from Association. In *CoRR cs. CL/0309034*.

J. Wiebe. 2000. Learning subjective adjectives from corpora. In *AAAI/IAAI*, pages 735–740.

Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis

Theresa Wilson
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
twilson@cs.pitt.edu

Janyce Wiebe
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
wiebe@cs.pitt.edu

Paul Hoffmann
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
hoffmanp@cs.pitt.edu

Abstract

This paper presents a new approach to phrase-level sentiment analysis that first determines whether an expression is neutral or polar and then disambiguates the polarity of the polar expressions. With this approach, the system is able to automatically identify the *contextual polarity* for a large subset of sentiment expressions, achieving results that are significantly better than baseline.

1 Introduction

Sentiment analysis is the task of identifying positive and negative opinions, emotions, and evaluations. Most work on sentiment analysis has been done at the document level, for example distinguishing positive from negative reviews. However, tasks such as multi-perspective question answering and summarization, opinion-oriented information extraction, and mining product reviews require sentence-level or even phrase-level sentiment analysis. For example, if a question answering system is to successfully answer questions about people's opinions, it must be able to pinpoint expressions of positive and negative sentiments, such as we find in the sentences below:

(1) African observers $generally\ approved^{+}$ of his victory while Western governments $denounced^{-}$ it.

(2) A succession of officers filled the TV screen to say they $supported^{+}$ the people and that the killings were "$not\ tolerable^{-}$."

(3) "We $don't\ hate^{+}$ the sinner," he says, "but we $hate^{-}$ the sin."

A typical approach to sentiment analysis is to start with a lexicon of positive and negative words and phrases. In these lexicons, entries are tagged with their a priori *prior polarity*: out of context, does the word seem to evoke something positive or something negative. For example, *beautiful* has a positive prior polarity, and *horrid* has a negative prior polarity. However, the *contextual polarity* of the phrase in which a word appears may be different from the word's prior polarity. Consider the underlined polarity words in the sentence below:

(4) Philip Clapp, president of the National Environment <u>Trust</u>, sums up <u>well</u> the general thrust of the reaction of environmental movements: "There is no <u>reason</u> at all to believe that the <u>polluters</u> are suddenly going to become <u>reasonable</u>."

Of these words, "Trust," "well," "reason," and "reasonable" have positive prior polarity, but they are not all being used to express positive sentiments. The word "reason" is negated, making the contextual polarity negative. The phrase "no reason at all to believe" changes the polarity of the proposition that follows; because "reasonable" falls within this proposition, its contextual polarity becomes negative. The word "Trust" is simply part of a referring expression and is not being used to express a sentiment; thus, its contextual polarity is neutral. Similarly for "polluters": in the context of the article, it simply refers to companies that pollute. Only "well" has the same prior and contextual polarity.

Many things must be considered in phrase-level sentiment analysis. Negation may be local (e.g., **not** *good*), or involve longer-distance dependencies such as the negation of the proposition (e.g., **does not** *look very good*) or the negation of the subject (e.g.,

no one *thinks that it's good*). In addition, certain phrases that contain negation words intensify rather than change polarity (e.g., **not only** *good but amazing*). Contextual polarity may also be influenced by modality (e.g., whether the proposition is asserted to be real (*realis*) or not real (*irrealis*) – *no reason at all to believe* is irrealis, for example); word sense (e.g., *Environmental* **Trust** versus *He has won the people's* **trust**); the syntactic role of a word in the sentence (e.g., **polluters** *are* versus *they are* **polluters**); and diminishers such as *little* (e.g., **little** *truth*, **little** *threat*). (See (Polanya and Zaenen, 2004) for a more detailed discussion of contextual polarity influencers.)

This paper presents new experiments in automatically distinguishing prior and contextual polarity. Beginning with a large stable of clues marked with prior polarity, we identify the contextual polarity of the phrases that contain instances of those clues in the corpus. We use a two-step process that employs machine learning and a variety of features. The first step classifies each phrase containing a clue as neutral or polar. The second step takes all phrases marked in step one as polar and disambiguates their contextual polarity (*positive*, *negative, both*, or *neutral*). With this approach, the system is able to automatically identify the contextual polarity for a large subset of sentiment expressions, achieving results that are significantly better than baseline. In addition, we describe new manual annotations of contextual polarity and a successful inter-annotator agreement study.

2 Manual Annotation Scheme

To create a corpus for the experiments below, we added contextual polarity judgments to existing annotations in the Multi-perspective Question Answering (MPQA) Opinion Corpus[1], namely to the annotations of *subjective expressions*[2]. A subjective expression is any word or phrase used to express an opinion, emotion, evaluation, stance, speculation, etc. A general covering term for such states is *private state* (Quirk et al., 1985). In the MPQA Corpus, subjective expressions of varying lengths are marked, from single words to long phrases.

For this work, our focus is on *sentiment expressions* – positive and negative expressions of emotions, evaluations, and stances. As these are types of subjective expressions, to create the corpus, we just needed to manually annotate the existing subjective expressions with their contextual polarity.

In particular, we developed an annotation scheme[3] for marking the contextual polarity of subjective expressions. Annotators were instructed to tag the polarity of subjective expressions as *positive*, *negative*, *both*, or *neutral*. The *positive* tag is for positive emotions (*I'm happy*), evaluations (*Great idea!*), and stances (*She supports the bill*). The *negative* tag is for negative emotions (*I'm sad*), evaluations (*Bad idea!*), and stances (*She's against the bill*). The *both* tag is applied to sentiment expressions that have both positive and negative polarity. The *neutral* tag is used for all other subjective expressions: those that express a different type of subjectivity such as speculation, and those that do not have positive or negative polarity.

Below are examples of contextual polarity annotations. The tags are in boldface, and the subjective expressions with the given tags are underlined.

(5) Thousands of coup supporters <u>celebrated</u> (**positive**) overnight, waving flags, blowing whistles ...

(6) The criteria set by Rice are the following: the three countries in question are <u>repressive</u> (**negative**) and <u>grave human rights violators</u> (**negative**) ...

(7) Besides, politicians refer to <u>good and evil</u> (**both**) only for purposes of intimidation and exaggeration.

(8) Jerome says the hospital <u>feels</u> (**neutral**) no different than a hospital in the states.

The annotators were asked to judge the contextual polarity of the sentiment that is ultimately being conveyed by the subjective expression, i.e., once the sentence has been fully interpreted. Thus, the subjective expression, *they have not succeeded, and*

[1]The MPQA Corpus is described in (Wiebe et al., 2005) and available at nrrc.mitre.org/NRRC/publications.htm.

[2]In the MPQA Corpus, subjective expressions are *direct subjective* expressions with *non-neutral expression intensity*, plus all the *expressive subjective elements*. Please see (Wiebe et al., 2005) for more details on the existing annotations in the MPQA Corpus.

[3]The annotation instructions are available at http://www.cs.pitt.edu/˜twilson.

will never succeed, was marked as positive in the sentence, *They have not succeeded, and will never succeed, in breaking the will of this valiant people*. The reasoning is that breaking the will of a valiant people is negative; hence, not succeeding in breaking their will is positive.

3 Agreement Study

To measure the reliability of the polarity annotation scheme, we conducted an agreement study with two annotators, using 10 documents from the MPQA Corpus. The 10 documents contain 447 subjective expressions. Table 1 shows the contingency table for the two annotators' judgments. Overall agreement is 82%, with a Kappa (κ) value of 0.72.

	Neutral	Positive	Negative	Both	Total
Neutral	123	14	24	0	161
Positive	16	73	5	2	96
Negative	14	2	167	1	184
Both	0	3	0	3	6
Total	153	92	196	6	447

Table 1: Agreement for Subjective Expressions (Agreement: 82%, κ: 0.72)

For 18% of the subjective expressions, at least one annotator used an *uncertain* tag when marking polarity. If we consider these cases to be borderline and exclude them from the study, percent agreement increases to 90% and Kappa rises to 0.84. Thus, the annotator agreement is especially high when both are certain. (Note that all annotations are included in the experiments described below.)

4 Corpus

In total, 15,991 subjective expressions from 425 documents (8,984 sentences) were annotated with contextual polarity as described above. Of these sentences, 28% contain no subjective expressions, 25% contain only one, and 47% contain two or more. Of the 4,247 sentences containing two or more subjective expressions, 17% contain mixtures of positive and negative expressions, and 62% contain mixtures of polar (positive/negative/both) and neutral subjective expressions.

The annotated documents are divided into two sets. The first (66 documents/1,373 sentences/2,808 subjective expressions) is a development set, used for data exploration and feature development. We use the second set (359 documents/7,611 sentences/13,183 subjective expressions) in 10-fold cross-validation experiments, described below.

5 Prior-Polarity Subjectivity Lexicon

For the experiments in this paper, we use a lexicon of over 8,000 *subjectivity clues*. Subjectivity clues are words and phrases that may be used to express private states, i.e., they have subjective usages (though they may have objective usages as well). For this work, only single-word clues are used.

To compile the lexicon, we began with a list of subjectivity clues from (Riloff and Wiebe, 2003). The words in this list were grouped in previous work according to their reliability as subjectivity clues. Words that are subjective in most contexts were marked strongly subjective (*strongsubj*), and those that may only have certain subjective usages were marked weakly subjective (*weaksubj*).

We expanded the list using a dictionary and a thesaurus, and also added words from the General Inquirer positive and negative word lists (General-Inquirer, 2000) which we judged to be potentially subjective. We also gave the new words reliability tags, either *strongsubj* or *weaksubj*.

The next step was to tag the clues in the lexicon with their prior polarity. For words that came from positive and negative word lists (General-Inquirer, 2000; Hatzivassiloglou and McKeown, 1997), we largely retained their original polarity, either *positive* or *negative*. We assigned the remaining words one of the tags *positive*, *negative*, *both* or *neutral*.

By far, the majority of clues, 92.8%, are marked as having either positive (33.1%) or negative (59.7%) prior polarity. Only a small number of clues (0.3%) are marked as having both positive and negative polarity. 6.9% of the clues in the lexicon are marked as neutral. Examples of these are verbs such as *feel*, *look*, and *think*, and intensifiers such as *deeply*, *entirely*, and *practically*. These words are included because, although their prior polarity is neutral, they are good clues that a sentiment is being expressed (e.g., **feels** *slighted*, **look** *forward to*). Including them increases the coverage of the system.

6 Experiments

The goal of the experiments described below is to classify the contextual polarity of the expressions that contain instances of the subjectivity clues in our lexicon. What the system specifically does is give each clue instance its own label. Note that the system does not try to identify expression boundaries. Doing so might improve performance and is a promising avenue for future research.

6.1 Definition of the Gold Standard

We define the gold standard used to train and test the system in terms of the manual annotations described in Section 2.

The gold standard class of a clue instance that is not in a subjective expression is *neutral*: since the clue is not even in a subjective expression, it is not contained in a sentiment expression.

Otherwise, if a clue instance appears in just one subjective expression (or in multiple subjective expressions with the same contextual polarity), then the class assigned to the clue instance is the class of the subjective expression(s). If a clue appears in at least one positive and one negative subjective expression (or in a subjective expression marked as *both*), then its class is *both*. If it is in a mixture of negative and neutral subjective expressions, its class is *negative*; if it is in a mixture of positive and neutral subjective expressions, its class is *positive*.

6.2 Performance of a Prior-Polarity Classifier

An important question is how useful prior polarity alone is for identifying contextual polarity. To answer this question, we create a classifier that simply assumes that the contextual polarity of a clue instance is the same as the clue's prior polarity, and we explore the classifier's performance on the development set.

This simple classifier has an accuracy of 48%. From the confusion matrix given in Table 2, we see that 76% of the errors result from words with non-neutral prior polarity appearing in phrases with neutral contextual polarity.

6.3 Contextual Polarity Disambiguation

The fact that words with non-neutral prior polarity so frequently appear in neutral contexts led us to adopt a two-step approach to contextual polarity disambiguation. For the first step, we concentrate on whether clue instances are neutral or polar in context (where *polar in context* refers to having a contextual polarity that is positive, negative or both). For the second step, we take all clue instances marked as polar in step one, and focus on identifying their contextual polarity. For both steps, we develop classifiers using the BoosTexter AdaBoost.HM (Schapire and Singer, 2000) machine learning algorithm with 5000 rounds of boosting. The classifiers are evaluated in 10-fold cross-validation experiments.

		Prior-Polarity Classifier				
		Neut	Pos	Neg	Both	Total
Gold	Neut	798	784	698	4	2284
	Pos	81	371	40	0	492
	Neg	149	181	622	0	952
	Both	4	11	13	5	33
	Total	1032	1347	1373	9	3761

Table 2: Confusion matrix for the prior-polarity classifier on the development set.

6.3.1 Neutral-Polar Classification

The neutral-polar classifier uses 28 features, listed in Table 3.

Word Features: *Word context* is a bag of three word tokens: the previous word, the word itself, and the next word. The *prior polarity* and *reliability class* are indicated in the lexicon.

Modification Features: These are binary relationship features. The first four involve relationships with the word immediately before or after: if the word is a noun preceded by an adjective, if the preceding word is an adverb other than *not*, if the preceding word is an intensifier, and if the word itself is an intensifier. A word is considered an intensifier if it appears in a list of intensifiers and if it precedes a word of the appropriate part-of-speech (e.g., an intensifier adjective must come before a noun).

The *modify* features involve the dependency parse tree for the sentence, obtained by first parsing the sentence (Collins, 1997) and then converting the tree into its dependency representation (Xia and Palmer, 2001). In a dependency representation, every node in the tree structure is a surface word (i.e., there are no abstract nodes such as NP or VP). The edge between a parent and a child specifies the grammatical relationship between the two words. Figure 1 shows

Word Features	Sentence Features	Structure Features
word token	strongsubj clues in current sentence: count	in subject: binary
word part-of-speech	strongsubj clues in previous sentence: count	in copular: binary
word context	strongsubj clues in next sentence: count	in passive: binary
prior polarity: positive, negative, both, neutral	weaksubj clues in current sentence: count	
reliability class: strongsubj or weaksubj	weaksubj clues in previous sentence: count	
Modification Features	weaksubj clues in next sentence: count	Document Feature
preceeded by adjective: binary	adjectives in sentence: count	document topic
preceeded by adverb (other than not): binary	adverbs in sentence (other than not): count	
preceeded by intensifier: binary	cardinal number in sentence: binary	
is intensifier: binary	pronoun in sentence: binary	
modifies strongsubj: binary	modal in sentence (other than will): binary	
modifies weaksubj: binary		
modified by strongsubj: binary		
modified by weaksubj: binary		

Table 3: Features for neutral-polar classification

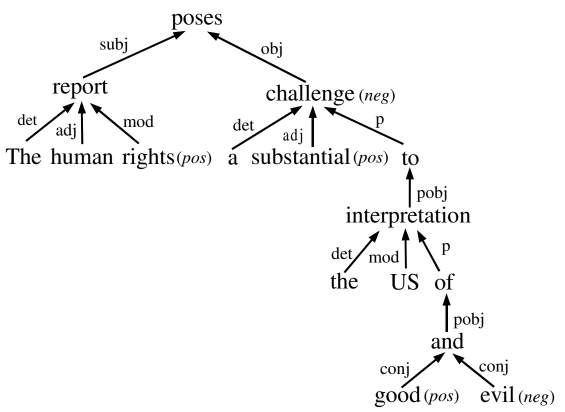


Figure 1: The dependency tree for the sentence *The human rights report poses a substantial challenge to the US interpretation of good and evil.* Prior polarity is marked in parentheses for words that match clues from the lexicon.

an example. The *modifies strongsubj/weaksubj* features are true if the word and its parent share an *adj*, *mod* or *vmod* relationship, and if its parent is an instance of a clue from the lexicon with strongsubj/weaksubj reliability. The *modified by strongsubj/weaksubj* features are similar, but look for relationships and clues in the word's children.

Structure Features: These are binary features that are determined by starting with the word instance and climbing up the dependency parse tree toward the root, looking for particular relationships, words, or patterns. The *in subject* feature is true if we find a *subj* relationship. The *in copular* feature is true if *in subject* is false and if a node along the path is both a main verb and a copular verb. The *in passive* features is true if a passive verb pattern is found on the climb.

Sentence Features: These are features that were found useful for sentence-level subjectivity classification by Wiebe and Riloff (2005). They include counts of strongsubj and weaksubj clues in the current, previous and next sentences, counts of adjectives and adverbs other than *not* in the current sentence, and binary features to indicate whether the sentence contains a pronoun, a cardinal number, and a modal other than *will*.

Document Feature: There is one document feature representing the topic of the document. A document may belong to one of 15 topics ranging from specific (e.g., the 2002 presidential election in Zimbabwe) to more general (e.g., economics) topics.

Table 4 gives neutral-polar classification results for the 28-feature classifier and two simpler classifiers that provide our baselines. The first row in the table lists the results for a classifier that uses just one feature, the word token. The second row shows the results for a classifier that uses both the word token and the word's prior polarity as features. The results for the 28-feature classifier are listed in the last row. The 28-feature classifier performs significantly better (1-tailed t-test, $p \leq .05$) than the two simpler classifiers, as measured by accuracy, polar F-measure, and neutral F-measure ($\beta = 1$). It has an accuracy of 75.9%, with a polar F-measure of 63.4 and a neutral F-measure of 82.1.

Focusing on the metrics for polar expressions, it's interesting to note that using just the word token as a feature produces a classifier with a precision slightly better than the 28-feature classifier, but with a recall that is 20% lower. Adding a feature for the prior

Word Features
word token
word prior polarity: positive, negative, both, neutral
Polarity Features
negated: binary
negated subject: binary
modifies polarity: positive, negative, neutral, both, notmod
modified by polarity: positive, negative, neutral, both, notmod
conj polarity: positive, negative, neutral, both, notmod
general polarity shifter: binary
negative polarity shifter: binary
positive polarity shifter: binary

Table 6: Features for polarity classification

polarity improves recall so that it is only 4.4% lower, but this hurts precision, which drops to 4.2% lower than the 28-feature classifier's precision. It is only with all the features that we get the best result, good precision with the highest recall.

The clues in the prior-polarity lexicon have 19,506 instances in the test set. According to the 28-feature neutral-polar classifier, 5,671 of these instances are polar in context. It is these clue instances that are passed on to the second step in the contextual disambiguation process, polarity classification.

6.3.2 Polarity Classification

Ideally, this second step in the disambiguation process would be a three-way classification task, determining whether the contextual polarity is positive, negative or both. However, although the majority of neutral expressions have been filtered out by the neutral-polar classification in step one, a number still remain. So, for this step, the polarity classification task remains four-way: positive, negative, both, and neutral.

Table 6 lists the features used by the polarity classifier. *Word token* and *word prior polarity* are unchanged from the neutral-polar classifier. *Negated* is a binary feature that captures whether the word is being locally negated: its value is true if a negation word or phrase is found within the four preceeding words or in any of the word's children in the dependency tree, and if the negation word is not in a phrase that intensifies rather than negates (e.g., *not only*). The *negated subject* feature is true if the subject of the clause containing the word is negated.

The *modifies polarity*, *modified by polarity*, and *conj polarity* features capture specific relationships between the word instance and other polarity words it may be related to. If the word and its parent in the dependency tree share an *obj*, *adj*, *mod*, or *vmod* relationship, the *modifies polarity* feature is set to the prior polarity of the word's parent (if the parent is not in our prior-polarity lexicon, its prior polarity is set to neutral). The *modified by polarity* feature is similar, looking for *adj*, *mod*, and *vmod* relationships and polarity clues within the word's children. The *conj polarity* feature determines if the word is in a conjunction. If so, the value of this feature is its sibling's prior polarity (as above, if the sibling is not in the lexicon, its prior polarity is neutral). Figure 1 helps to illustrate these features: *modifies polarity* is negative for the word "substantial," *modified by polarity* is positive for the word "challenge," and *conj polarity* is negative for the word "good."

The last three polarity features look in a window of four words before, searching for the presence of particular types of polarity influencers. *General polarity shifters* reverse polarity (e.g., *little* truth, *little* threat). *Negative polarity shifters* typically make the polarity of an expression negative (e.g., *lack* of understanding). *Positive polarity shifters* typically make the polarity of an expression positive (e.g., *abate* the damage).

The polarity classification results for this second step in the contextual disambiguation process are given in Table 5. Also listed in the table are results for the two simple classifiers that provide our baselines. The first line in Table 5 lists the results for the classifier that uses just one feature, the word token. The second line shows the results for the classifier that uses both the word token and the word's prior polarity as features. The last line shows the results for the polarity classifier that uses all 10 features from Table 6.

Mirroring the results from step one, the more complex classifier performs significantly better than the simpler classifiers, as measured by accuracy and all of the F-measures. The 10-feature classifier achieves an accuracy of 65.7%, which is 4.3% higher than the more challenging baseline provided by the word + prior polarity classifier. Positive F-measure is 65.1 (5.7% higher); negative F-measure is 77.2 (2.3% higher); and neutral F-measure is 46.2 (13.5% higher).

Focusing on the metrics for positive and negative expressions, we again see that the simpler classifiers

	Acc	Polar Rec	Polar Prec	Polar F	Neut Rec	Neut Prec	Neut F
word token	73.6	45.3	72.2	55.7	89.9	74.0	81.2
word+priorpol	74.2	54.3	68.6	60.6	85.7	76.4	80.7
28 features	75.9	56.8	71.6	63.4	87.0	77.7	82.1

Table 4: Results for Step 1 Neutral-Polar Classification

		Positive			Negative			Both			Neutral		
	Acc	Rec	Prec	F	Rec	Prec	F	Rec	Prec	F	Rec	Prec	F
word token	61.7	59.3	63.4	61.2	83.9	64.7	73.1	9.2	35.2	14.6	30.2	50.1	37.7
word+priorpol	63.0	69.4	55.3	61.6	80.4	71.2	75.5	9.2	35.2	14.6	33.5	51.8	40.7
10 features	65.7	67.1	63.3	65.1	82.1	72.9	77.2	11.2	28.4	16.1	41.4	52.4	46.2

Table 5: Results for Step 2 Polarity Classification.

Experiment	Features Removed
AB1	negated, negated subject
AB2	modifies polarity, modified by polarity
AB3	conj polarity
AB4	general, negative, and positive polarity shifters

Table 7: Features for polarity classification

take turns doing better or worse for precision and recall. Using just the word token, positive precision is slightly higher than for the 10-feature classifier, but positive recall is 11.6% lower. Add the prior polarity, and positive recall improves, but at the expense of precision, which is 12.6% lower than for the 10-feature classifier. The results for negative expressions are similar. The word-token classifier does well on negative recall but poorly on negative precision. When prior polarity is added, negative recall improves but negative precision drops. It is only with the addition of the polarity features that we achieve both higher precisions and higher recalls.

To explore how much the various polarity features contribute to the performance of the polarity classifier, we perform four experiments. In each experiment, a different set of polarity features is excluded, and the polarity classifier is retrained and evaluated. Table 7 lists the features that are removed for each experiment.

The only significant difference in performance in these experiments is neutral F-measure when the modification features (AB2) are removed. These ablation experiments show that the combination of features is needed to achieve significant results over baseline for polarity classification.

7 Related Work

Much work on sentiment analysis classifies documents by their overall sentiment, for example determining whether a review is positive or negative (e.g., (Turney, 2002; Dave et al., 2003; Pang and Lee, 2004; Beineke et al., 2004)). In contrast, our experiments classify individual words and phrases. A number of researchers have explored learning words and phrases with *prior* positive or negative polarity (another term is *semantic orientation*) (e.g., (Hatzivassiloglou and McKeown, 1997; Kamps and Marx, 2002; Turney, 2002)). In contrast, we begin with a lexicon of words with established prior polarities, and identify the *contextual polarity* of phrases in which instances of those words appear in the corpus. To make the relationship between that task and ours clearer, note that some word lists used to evaluate methods for recognizing prior polarity are included in our prior-polarity lexicon (General Inquirer lists (General-Inquirer, 2000) used for evaluation by Turney, and lists of manually identified positive and negative adjectives, used for evaluation by Hatzivassiloglou and McKeown).

Some research classifies the sentiments of sentences. Yu and Hatzivassiloglou (2003), Kim and Hovy (2004), Hu and Liu (2004), and Grefenstette et al. (2001)[4] all begin by first creating prior-polarity lexicons. Yu and Hatzivassiloglou then assign a sentiment to a sentence by averaging the prior semantic orientations of instances of lexicon words in the sentence. Thus, they do not identify the contextual polarity of individual phrases containing clues, as we

[4]In (Grefenstette et al., 2001), the units that are classified are fixed windows around named entities rather than sentences.

do in this paper. Kim and Hovy, Hu and Liu, and Grefenstette et al. multiply or count the prior polarities of clue instances in the sentence. They also consider local negation to reverse polarity. However, they do not use the other types of features in our experiments, and they restrict their tags to *positive* and *negative* (excluding our *both* and *neutral* categories). In addition, their systems assign one sentiment per sentence; our system assigns contextual polarity to individual expressions. As seen above, sentences often contain more than one sentiment expression.

Nasukawa, Yi, and colleagues (Nasukawa and Yi, 2003; Yi et al., 2003) classify the contextual polarity of sentiment expressions, as we do. Thus, their work is probably most closely related to ours. They classify expressions that are about specific items, and use manually developed patterns to classify polarity. These patterns are high-quality, yielding quite high precision, but very low recall. Their system classifies a much smaller proportion of the sentiment expressions in a corpus than ours does.

8 Conclusions

In this paper, we present a new approach to phrase-level sentiment analysis that first determines whether an expression is neutral or polar and then disambiguates the polarity of the polar expressions. With this approach, we are able to automatically identify the *contextual polarity* for a large subset of sentiment expressions, achieving results that are significantly better than baseline.

9 Acknowledgments

This work was supported in part by the NSF under grant IIS-0208798 and by the Advanced Research and Development Activity (ARDA).

References

P. Beineke, T. Hastie, and S. Vaithyanathan. 2004. The sentimental factor: Improving review classification via human-provided information. In *ACL-2004*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL-1997*.

K. Dave, S. Lawrence, and D. M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW-2003*.

The General-Inquirer. 2000. *http://www.wjh.harvard.edu/˜inquirer/spreadsheet_guide.htm*.

G. Grefenstette, Y. Qu, J.G. Shanahan, and D.A. Evans. 2001. Coupling niche browsers and affect analysis for an opinion mining application. In *RIAO-2004*.

V. Hatzivassiloglou and K. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL-1997*.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *KDD-2004*.

J. Kamps and M. Marx. 2002. Words with attitude. In *1st International WordNet Conference*.

S-M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *Coling 2004*.

T. Nasukawa and J. Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *K-CAP 2003*.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL-2004*.

L. Polanya and A. Zaenen. 2004. Contextual valence shifters. In *Working Notes — Exploring Attitude and Affect in Text (AAAI Spring Symposium Series)*.

R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language.* Longman, New York.

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *EMNLP-2003*.

R. E. Schapire and Y. Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

P. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *ACL-2002*.

J. Wiebe and E. Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *CICLing-2005*.

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evalution (formerly Computers and the Humanities)*, 1(2).

F. Xia and M. Palmer. 2001. Converting dependency structures to phrase structures. In *HLT-2001*.

J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *IEEE ICDM-2003*.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *EMNLP-2003*.

Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns

Yejin Choi and Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY 14853
{ychoi,cardie}@cs.cornell.edu

Ellen Riloff and Siddharth Patwardhan
School of Computing
University of Utah
Salt Lake City, UT 84112
{riloff,sidd}@cs.utah.edu

Abstract

Recent systems have been developed for sentiment classification, opinion recognition, and opinion analysis (e.g., detecting polarity and strength). We pursue another aspect of opinion analysis: identifying the sources of opinions, emotions, and sentiments. We view this problem as an information extraction task and adopt a hybrid approach that combines Conditional Random Fields (Lafferty et al., 2001) and a variation of AutoSlog (Riloff, 1996a). While CRFs model source identification as a sequence tagging task, AutoSlog learns extraction patterns. Our results show that the combination of these two methods performs better than either one alone. The resulting system identifies opinion sources with 79.3% precision and 59.5% recall using a head noun matching measure, and 81.2% precision and 60.6% recall using an overlap measure.

1 Introduction

In recent years, there has been a great deal of interest in methods for automatically identifying opinions, emotions, and sentiments in text. Much of this research explores sentiment classification, a text categorization task in which the goal is to classify a document as having positive or negative polarity (e.g., Das and Chen (2001), Pang et al. (2002), Turney (2002), Dave et al. (2003), Pang and Lee (2004)). Other research efforts analyze opinion expressions at the sentence level or below to recognize opinions, their polarity, and their strength (e.g., Dave et al. (2003), Pang and Lee (2004), Wilson et al. (2004), Yu and Hatzivassiloglou (2003), Wiebe and Riloff (2005)). Many applications could benefit from these opinion analyzers, including product reputation tracking (e.g., Morinaga et al. (2002), Yi et al. (2003)), opinion-oriented summarization (e.g., Cardie et al. (2004)), and question answering (e.g., Bethard et al. (2004), Yu and Hatzivassiloglou (2003)).

We focus here on another aspect of opinion analysis: automatically identifying the sources of the opinions. Identifying opinion sources will be especially critical for opinion-oriented question-answering systems (e.g., systems that answer questions of the form "How does [X] feel about [Y]?") and opinion-oriented summarization systems, both of which need to distinguish the opinions of one source from those of another.[1]

The goal of our research is to identify direct and indirect sources of opinions, emotions, sentiments, and other *private states* that are expressed in text. To illustrate the nature of this problem, consider the examples below:

S1: <u>Taiwan-born voters</u> favoring independence...

[1]In related work, we investigate methods to identify the opinion expressions (e.g., Riloff and Wiebe (2003), Wiebe and Riloff (2005), Wilson et al. (2005)) and the nesting structure of sources (e.g., Breck and Cardie (2004)). The *target* of each opinion, i.e., what the opinion is directed towards, is currently being annotated manually for our corpus.

S2: According to the report, the human rights record in China is horrendous.

S3: International officers believe that the EU will prevail.

S4: International officers said US officials want the EU to prevail.

In S1, the phrase "Taiwan-born voters" is the direct (i.e., first-hand) source of the "favoring" sentiment. In S2, "the report" is the direct source of the opinion about China's human rights record. In S3, "International officers" are the direct source of an opinion regarding the EU. The same phrase in S4, however, denotes an indirect (i.e., second-hand, third-hand, etc.) source of an opinion whose direct source is "US officials".

In this paper, we view *source identification* as an information extraction task and tackle the problem using sequence tagging and pattern matching techniques simultaneously. Using syntactic, semantic, and orthographic lexical features, dependency parse features, and opinion recognition features, we train a linear-chain Conditional Random Field (CRF) (Lafferty et al., 2001) to identify opinion sources. In addition, we employ features based on automatically learned extraction patterns and perform feature induction on the CRF model.

We evaluate our hybrid approach using the NRRC corpus (Wiebe et al., 2005), which is manually annotated with direct and indirect opinion source information. Experimental results show that the CRF model performs well, and that both the extraction patterns and feature induction produce performance gains. The resulting system identifies opinion sources with 79.3% precision and 59.5% recall using a head noun matching measure, and 81.2% precision and 60.6% recall using an overlap measure.

2 The Big Picture

The goal of information extraction (IE) systems is to extract information about events, including the participants of the events. This task goes beyond Named Entity recognition (e.g., Bikel et al. (1997)) because it requires the recognition of role relationships. For example, an IE system that extracts information about corporate acquisitions must distinguish between the company that is doing the acquiring and the company that is being acquired. Similarly, an IE system that extracts information about terrorism must distinguish between the person who is the perpetrator and the person who is the victim. We hypothesized that IE techniques would be well-suited for source identification because an opinion statement can be viewed as a kind of speech event with the source as the agent.

We investigate two very different learning-based methods from information extraction for the problem of opinion source identification: graphical models and extraction pattern learning. In particular, we consider Conditional Random Fields (Lafferty et al., 2001) and a variation of AutoSlog (Riloff, 1996a). CRFs have been used successfully for Named Entity recognition (e.g., McCallum and Li (2003), Sarawagi and Cohen (2004)), and AutoSlog has performed well on information extraction tasks in several domains (Riloff, 1996a). While CRFs treat source identification as a sequence tagging task, AutoSlog views the problem as a pattern-matching task, acquiring symbolic patterns that rely on both the syntax and lexical semantics of a sentence. We hypothesized that a combination of the two techniques would perform better than either one alone.

Section 3 describes the CRF approach to identifying opinion sources and the features that the system uses. Section 4 then presents a new variation of AutoSlog, *AutoSlog-SE*, which generates IE patterns to extract sources. Section 5 describes the hybrid system: we encode the IE patterns as additional features in the CRF model. Finally, Section 6 presents our experimental results and error analysis.

3 Semantic Tagging via Conditional Random Fields

We defined the problem of opinion source identification as a sequence tagging task via CRFs as follows. Given a sequence of tokens, $x = x_1 x_2 ... x_n$, we need to generate a sequence of tags, or labels, $y = y_1 y_2 ... y_n$. We define the set of possible label values as `'S'`, `'T'`, `'-'`, where `'S'` is the first token (or Start) of a source, `'T'` is a non-initial token (i.e., a conTinuation) of a source, and `'-'` is a token that is not part of any source.[2]

A detailed description of CRFs can be found in

[2]This is equivalent to the IOB tagging scheme used in syntactic chunkers (Ramshaw and Marcus, 1995).

Lafferty et al. (2001). For our sequence tagging problem, we create a linear-chain CRF based on an undirected graph $G = (V, E)$, where V is the set of random variables $Y = \{Y_i | 1 \leq i \leq n\}$, one for each of n tokens in an input sentence; and $E = \{(Y_{i-1}, Y_i) | 1 < i \leq n\}$ is the set of $n - 1$ edges forming a linear chain. For each sentence x, we define a non-negative clique potential $\exp(\sum_{k=1}^{K} \lambda_k f_k(y_{i-1}, y_i, x))$ for each edge, and $\exp(\sum_{k=1}^{K'} \lambda'_k f'_k(y_i, x))$ for each node, where $f_k(...)$ is a binary feature indicator function, λ_k is a weight assigned for each feature function, and K and K' are the number of features defined for edges and nodes respectively. Following Lafferty et al. (2001), the conditional probability of a sequence of labels y given a sequence of tokens x is:

$$P(y|x) = \frac{1}{Z_x} \exp\left(\sum_{i,k} \lambda_k\, f_k(y_{i-1}, y_i, x) + \sum_{i,k} \lambda'_k\, f'_k(y_i, x)\right) \tag{1}$$

$$Z_x = \sum_{y} \exp\left(\sum_{i,k} \lambda_k\, f_k(y_{i-1}, y_i, x) + \sum_{i,k} \lambda'_k\, f'_k(y_i, x)\right) \tag{2}$$

where Z_x is a normalization constant for each x. Given the training data D, a set of sentences paired with their correct `'ST-'` source label sequences, the parameters of the model are trained to maximize the conditional log-likelihood $\prod_{(x,y) \in D} P(y|x)$. For inference, given a sentence x in the test data, the tagging sequence y is given by $\text{argmax}_{y'} P(y'|x)$.

3.1 Features

To develop features, we considered three properties of opinion sources. First, the sources of opinions are mostly noun phrases. Second, the source phrases should be semantic entities that can bear or express opinions. Third, the source phrases should be directly related to an opinion expression. When considering only the first and second criteria, this task reduces to named entity recognition. Because of the third condition, however, the task requires the recognition of opinion expressions and a more sophisticated encoding of sentence structure to capture relationships between source phrases and opinion expressions.

With these properties in mind, we define the following features for each token/word x_i in an input sentence. For pedagogical reasons, we will describe some of the features as being multi-valued or categorical features. In practice, however, all features are binarized for the CRF model.

Capitalization features We use two boolean features to represent the capitalization of a word: `all-capital`, `initial-capital`.

Part-of-speech features Based on the lexical categories produced by GATE (Cunningham et al., 2002), each token x_i is classified into one of a set of coarse part-of-speech tags: noun, verb, adverb, wh-word, determiner, punctuation, etc. We do the same for neighboring words in a $[-2, +2]$ window in order to assist noun phrase segmentation.

Opinion lexicon features For each token x_i, we include a binary feature that indicates whether or not the word is in our *opinion lexicon* — a set of words that indicate the presence of an opinion. We do the same for neighboring words in a $[-1, +1]$ window. Additionally, we include for x_i a feature that indicates the opinion subclass associated with x_i, if available from the lexicon. (e.g., "bless" is classified as "moderately subjective" according to the lexicon, while "accuse" and "berate" are classified more specifically as "judgments".) The lexicon is initially populated with approximately 500 opinion words [3] from (Wiebe et al., 2002), and then augmented with opinion words identified in the training data. The training data contains manually produced phrase-level annotations for all expressions of opinions, emotions, etc. (Wiebe et al., 2005). We collected all content words that occurred in the training set such that at least 50% of their occurrences were in opinion annotations.

Dependency tree features For each token x_i, we create features based on the parse tree produced by the Collins (1999) dependency parser. The purpose of the features is to (1) encode structural information, and (2) indicate whether x_i is involved in any grammatical relations with an opinion word. Two pre-processing steps are required before features can be constructed:

[3] Some words are drawn from Levin (1993); others are from Framenet lemmas (Baker et al. 1998) associated with communication verbs.

1. **Syntactic chunking.** We traverse the dependency tree using breadth-first search to identify and group syntactically related nodes, producing a flatter, more concise tree. Each syntactic "chunk" is also assigned a grammatical role (e.g., `subject`, `object`, `verb modifier`, `time`, `location`, `of-pp`, `by-pp`) based on its constituents. Possessives (e.g., "Clinton's idea") and the phrase "according to X" are handled as special cases in the chunking process.

2. **Opinion word propagation.** Although the opinion lexicon contains only content words and no multi-word phrases, actual opinions often comprise an entire phrase, e.g., *"is really willing"* or *"in my opinion"*. As a result, we mark as an opinion the entire chunk that contains an opinion word. This allows each token in the chunk to act as an opinion word for feature encoding.

After syntactic chunking and opinion word propagation, we create the following dependency tree features for each token x_i:

- the grammatical role of its chunk
- the grammatical role of x_{i-1}'s chunk
- whether the parent chunk includes an opinion word
- whether x_i's chunk is in an argument position with respect to the parent chunk
- whether x_i represents a constituent boundary

Semantic class features We use 7 binary features to encode the semantic class of each word x_i: `authority`, `government`, `human`, `media`, `organization_or_company`, `proper_name`, and `other`. The `other` class captures 13 semantic classes that cannot be sources, such as `vehicle` and `time`.

Semantic class information is derived from named entity and semantic class labels assigned to x_i by the Sundance shallow parser (Riloff, 2004). Sundance uses named entity recognition rules to label noun phrases as belonging to named entity classes, and assigns semantic tags to individual words based on a semantic dictionary. Table 1 shows the hierarchy that Sundance uses for semantic classes associated with opinion sources. Sundance is also used to recognize and instantiate the source extraction patterns that are learned by AutoSlog-SE, which is described in the next section.

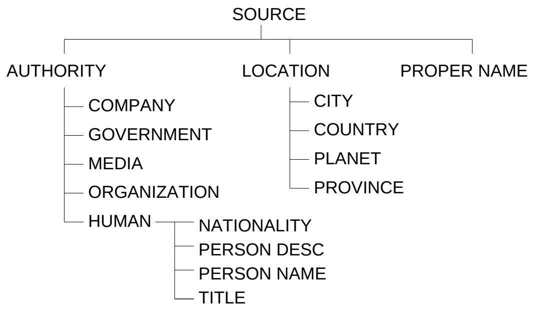


Figure 1: The semantic hierarchy for opinion sources

4 Semantic Tagging via Extraction Patterns

We also learn patterns to extract opinion sources using a statistical adaptation of the AutoSlog IE learning algorithm. AutoSlog (Riloff, 1996a) is a supervised extraction pattern learner that takes a training corpus of texts and their associated answer keys as input. A set of heuristics looks at the context surrounding each answer and proposes a lexico-syntactic pattern to extract that answer from the text. The heuristics are not perfect, however, so the resulting set of patterns needs to be manually reviewed by a person.

In order to build a fully automatic system that does not depend on manual review, we combined AutoSlog's heuristics with statistics from the annotated training data to create a fully automatic supervised learner. We will refer to this learner as AutoSlog-SE (Statistically Enhanced variation of AutoSlog). AutoSlog-SE's learning process has three steps:

Step 1: AutoSlog's heuristics are applied to every noun phrase (NP) in the training corpus. This generates a set of extraction patterns that, collectively, can extract every NP in the training corpus.

Step 2: The learned patterns are augmented with selectional restrictions that semantically constrain the types of noun phrases that are legitimate extractions for opinion sources. We used the semantic classes shown in Figure 1 as selectional restrictions.

Step 3: The patterns are applied to the training corpus and statistics are gathered about their extractions. We count the number of extractions that match annotations in the corpus (correct extractions) and the number of extractions that do not match annotations (incorrect extractions). These counts are then used to estimate the probability that the pattern will extract an opinion source in new texts:

$$P(\text{source} \mid \text{pattern}_i) = \frac{\text{correct sources}}{\text{correct sources} + \text{incorrect sources}}$$

This learning process generates a set of extraction patterns coupled with probabilities. In the next section, we explain how these extraction patterns are represented as features in the CRF model.

5 Extraction Pattern Features for the CRF

The extraction patterns provide two kinds of information. `SourcePatt` indicates whether a word activates any source extraction pattern. For example, the word *"complained"* activates the pattern *"<subj> complained"* because it anchors the expression. `SourceExtr` indicates whether a word is extracted by any source pattern. For example, in the sentence "President Jacques Chirac frequently complained about France's economy", the words "President", "Jacques", and "Chirac" would all be extracted by the *"<subj> complained"* pattern.

Each extraction pattern has frequency and probability values produced by AutoSlog-SE, hence we create four IE pattern-based features for each token x_i: `SourcePatt-Freq`, `SourceExtr-Freq`, `SourcePatt-Prob`, and `SourceExtr-Prob`, where the frequency values are divided into three ranges: {0, 1, 2+} and the probability values are divided into five ranges of equal size.

6 Experiments

We used the Multi-Perspective Question Answering (MPQA) corpus[4] for our experiments. This corpus consists of 535 documents that have been manually annotated with opinion-related information including direct and indirect sources. We used 135 documents as a tuning set for model development and feature engineering, and used the remaining 400 documents for evaluation, performing 10-fold cross validation. These texts are English language versions of articles that come from many countries and cover many topics.[5]

We evaluate performance using 3 measures: overlap match (OL), head match (HM), and exact match (EM). OL is a lenient measure that considers an extraction to be correct if it overlaps with any of the annotated words. HM is a more conservative measure that considers an extraction to be correct if its head matches the head of the annotated source. We report these somewhat loose measures because the annotators vary in where they place the exact boundaries of a source. EM is the strictest measure that requires an exact match between the extracted words and the annotated words. We use three evaluation metrics: recall, precision, and F-measure with recall and precision equally weighted.

6.1 Baselines

We developed three baseline systems to assess the difficulty of our task. *Baseline-1* labels as sources all phrases that belong to the semantic categories `authority`, `government`, `human`, `media`, `organization_or_company`, `proper_name`. Table 1 shows that the precision is poor, suggesting that the third condition described in Section 3.1 (opinion recognition) does play an important role in source identification. The recall is much higher but still limited due to sources that fall outside of the semantic categories or are not recognized as belonging to these categories. *Baseline-2* labels a noun phrase as a source if any of the following are true: (1) the NP is the subject of a verb phrase containing an opinion word, (2) the NP follows *"according to"*, (3) the NP contains a possessive and is preceded by an opinion word, or (4) the NP follows *"by"* and attaches to an opinion word. *Baseline-2*'s heuristics are designed to address the first and the third conditions in Section 3.1. Table 1 shows that *Baseline-2* is substantially better than *Baseline-1*. *Baseline-3*

[4]The MPQA corpus can be freely obtained at http://nrrc.mitre.org/NRRC/publications.htm.

[5]This data was obtained from the Foreign Broadcast Information Service (FBIS), a U.S. government agency.

		Recall	Prec	F1
Baseline-1	OL	77.3	28.8	42.0
	HM	71.4	28.6	40.8
	EM	65.4	20.9	31.7
Baseline-2	OL	62.4	60.5	61.4
	HM	59.7	58.2	58.9
	EM	50.8	48.9	49.8
Baseline-3	OL	49.9	72.6	59.2
	HM	47.4	72.5	57.3
	EM	44.3	58.2	50.3
Extraction Patterns	OL	48.5	81.3	60.8
	HM	46.9	78.5	58.7
	EM	41.9	70.2	52.5
CRF: basic features	OL	56.1	81.0	66.3
	HM	55.1	79.2	65.0
	EM	50.0	72.4	59.2
CRF: basic + IE pattern features	OL	59.1	82.4	68.9
	HM	58.1	80.5	67.5
	EM	52.5	73.3	61.2
CRF-FI: basic features	OL	57.7	80.7	67.3
	HM	56.8	78.8	66.0
	EM	51.7	72.4	60.3
CRF-FI: basic + IE pattern features	OL	60.6	81.2	69.4
	HM	59.5	79.3	68.0
	EM	54.1	72.7	62.0

Table 1: Source identification performance table

labels a noun phrase as a source if it satisfies both *Baseline-1* and *Baseline-2*'s conditions (this should satisfy all three conditions described in Section 3.1). As shown in Table 1, the precision of this approach is the best of the three baselines, but the recall is the lowest.

6.2 Extraction Pattern Experiment

We evaluated the performance of the learned extraction patterns on the source identification task. The learned patterns were applied to the test data and the extracted sources were scored against the manual annotations.[6] Table 1 shows that the extraction patterns produced lower recall than the baselines, but with considerably higher precision. These results show that the extraction patterns alone can identify nearly half of the opinion sources with good accuracy.

[6]These results were obtained using the patterns that had a probability $> .50$ and frequency > 1.

6.3 CRF Experiments

We developed our CRF model using the MALLET code from McCallum (2002). For training, we used a Gaussian prior of 0.25, selected based on the tuning data. We evaluate the CRF using the *basic features* from Section 3, both with and without the IE pattern features from Section 5. Table 1 shows that the CRF with basic features outperforms all of the baselines as well as the extraction patterns, achieving an F-measure of 66.3 using the OL measure, 65.0 using the HM measure, and 59.2 using the EM measure. Adding the IE pattern features further increases performance, boosting recall by about 3 points for all of the measures and slightly increasing precision as well.

CRF with feature induction. One limitation of log-linear function models like CRFs is that they cannot form a decision boundary from conjunctions of existing features, unless conjunctions are explicitly given as part of the feature vector. For the task of identifying opinion sources, we observed that the model could benefit from conjunctive features. For instance, instead of using two separate features, HUMAN and PARENT-CHUNK-INCLUDES-OPINION-EXPRESSION, the conjunction of the two is more informative.

For this reason, we applied the CRF feature induction approach introduced by McCallum (2003). As shown in Table 1, where CRF-FI stands for the CRF model with feature induction, we see consistent improvements by automatically generating conjunctive features. The final system, which combines the basic features, the IE pattern features, and feature induction achieves an F-measure of 69.4 (recall=60.6%, precision=81.2%) for the OL measure, an F-measure of 68.0 (recall=59.5%, precision=79.3%) for the HM measure, and an F-measure of 62.0 (recall=54.1%, precision=72.7%) for the EM measure.

6.4 Error Analysis

An analysis of the errors indicated some common mistakes:

- Some errors resulted from error propagation in our subsystems. Errors from the sentence boundary detector in GATE (Cunningham et al., 2002) were especially problematic because they caused the Collins parser to fail, resulting in no dependency tree information.

• Some errors were due to complex and unusual sentence structure, which our rather simple feature encoding for CRF could not capture well.

• Some errors were due to the limited coverage of the opinion lexicon. We failed to recognize some cases when idiomatic or vague expressions were used to express opinions.

Below are some examples of errors that we found interesting. Doubly underlined phrases indicate incorrectly extracted sources (either false positives or false negatives). Opinion words are singly underlined.

False positives:

(1) Actually, these three countries do have one common denominator, i.e., that their values and policies do not agree with those of the United States and none of them are on good terms with the United States.

(2) Perhaps this is why Fidel Castro has not spoken out against what might go on in Guantanamo.

In (1), "their values and policies" seems like a reasonable phrase to extract, but the annotation does not mark this as a source, perhaps because it is somewhat abstract. In (2), "spoken out" is negated, which means that the verb phrase does not bear an opinion, but our system failed to recognize the negation.

False negatives:

(3) And for this reason, too, they have a moral duty to speak out, as Swedish Foreign Minister Anna Lindh, among others, did yesterday.

(4) In particular, Iran and Iraq are at loggerheads with each other to this day.

Example (3) involves a complex sentence structure that our system could not deal with. (4) involves an uncommon opinion expression that our system did not recognize.

7 Related Work

To our knowledge, our research is the first to automatically identify opinion sources using the MPQA opinion annotation scheme. The most closely related work on opinion analysis is Bethard et al. (2004), who use machine learning techniques to identify propositional opinions and their holders (sources). However, their work is more limited in scope than ours in several ways. Their work only addresses propositional opinions, which are "localized in the propositional argument" of certain verbs such as "believe" or "realize". In contrast, our work aims to find sources for all opinions, emotions, and sentiments, including those that are not related to a verb at all. Furthermore, Berthard et al.'s task definition only requires the identification of direct sources, while our task requires the identification of both direct and indirect sources. Bethard et al. evaluate their system on manually annotated FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) sentences and achieve 48% recall with 57% precision.

Our IE pattern learner can be viewed as a cross between AutoSlog (Riloff, 1996a) and AutoSlog-TS (Riloff, 1996b). AutoSlog is a supervised learner that requires annotated training data but does not compute statistics. AutoSlog-TS is a weakly supervised learner that does not require annotated data but generates coarse statistics that measure each pattern's correlation with relevant and irrelevant documents. Consequently, the patterns learned by both AutoSlog and AutoSlog-TS need to be manually reviewed by a person to achieve good accuracy. In contrast, our IE learner, AutoSlog-SE, computes statistics directly from the annotated training data, creating a fully automatic variation of AutoSlog.

8 Conclusion

We have described a hybrid approach to the problem of extracting sources of opinions in text. We cast this problem as an information extraction task, using both CRFs and extraction patterns. Our research is the first to identify both direct and indirect sources for all types of opinions, emotions, and sentiments.

Directions for future work include trying to increase recall by identifying relationships between opinions and sources that cross sentence boundaries, and relationships between multiple opinion expressions by the same source. For example, the fact that a coreferring noun phrase was marked as a source in one sentence could be a useful clue for extracting the source from another sentence. The probability or the strength of an opinion expression may also play a useful role in encouraging or suppressing source extraction.

9 Acknowledgments

We thank the reviewers for their many helpful comments, and the Cornell NLP group for their advice and suggestions for improvement. This work was supported by the Advanced Research and Development Activity (ARDA), by NSF Grants IIS-0208028 and IIS-0208985, and by the Xerox Foundation.

References

C. Baker, C. Fillmore & J. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the COLING-ACL*.

S. Bethard, H. Yu, A. Thornton, V. Hativassiloglou & D. Jurafsky. 2004. Automatic extraction of opinion propositions and their holders. In *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text*.

D. Bikel, S. Miller, R. Schwartz & R. Weischedel. 1997. Nymble: A High-Performance Learning Name-Finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing.*

E. Breck & C. Cardie. 2004. Playing the Telephone Game: Determining the Hierarchical Structure of Perspective and Speech Expressions. In *Proceedings of 20th International Conference on Computational Linguistics*.

C. Cardie, J. Wiebe, T. Wilson & D. Litman. 2004. Low-level annotations and summary representations of opinions for multiperspective QA. In *New Directions in Question Answering*. AAAI Press/MIT Press.

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

H. Cunningham, D. Maynard, K. Bontcheva & V. Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

S. Das & M. Chen. 2001. Yahoo for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the 8th Asia Pacific Finance Association Annual Conference*.

K. Dave, S. Lawrence & D. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classifi cation of product reviews. In *International World Wide Web Conference*.

J. Lafferty, A. K. McCallum & F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of 18th International Conference on Machine Learning*.

B. Levin. 1993. English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press.

A. K. McCallum. 2002. *MALLET: A Machine Learning for Language Toolkit.* http://mallet.cs.umass.edu.

A. K. McCallum. 2003. Effi ciently Inducing Features of Conditional Random Fields. In *Conference on Uncertainty in Artificial Intelligence*.

A. K. McCallum & W. Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In *Conference on Natural Language Learning.*

S. Morinaga, K. Yamanishi, K. Tateishi & T. Fukushima 2002. Mining Product Reputations on the Web. In *Proceedings of the 8th Internatinal Conference on Knowledge Discover and Data Mining*.

M. Palmer, D. Gildea & P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. In *Computational Linguistics* 31.

B. Pang, L. Lee & S. Vaithyanathan. 2002. Thumbs up? sentiment classifi cation using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

B. Pang & L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

L. A. Ramshaw & M. P. Marcus. 1995. Nymble: A High-Performance Learning Name-Finder. In *Proceedings of the 3rd Workshop on Very Large Corpora*.

E. Riloff. 1996a. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. In *Artificial Intelligence*, Vol. 85.

E. Riloff. 1996b. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the 13th National Conference on Artificial Intelligence.*

E. Riloff & J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceesings of 2003 Conference on Empirical Methods in Natural Language Processing*.

E. Riloff & W. Phillips. 2004. An Introduction to the Sundance and AutoSlog Systems Technical Report UUCS-04-015, School of Computing, University of Utah.

S. Sarawagi & W. W. Cohen. 2004. Semi-Markov Conditional Random Fields for Information Extraction *18th Annual Conference on Neural Information Processing Systems*.

P. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classifi cation of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

T. Wilson, J. Wiebe & R. Hwa. 2004. Just how mad are you? fi nding strong and weak opinion clauses. In *Proceedings of the 9th National Conference on Artificial Intelligence*.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff & S. Patwardhan. 2005. OpinionFinder: A system for subjectivity analysis. Demonstration Description in *Conference on Empirical Methods in Natural Language Processing.*

J. Yi, T. Nasukawa, R. Bunescu & W. Niblack. 2003. Sentiment Analyzer: Extracting Sentiments about a Given Topic using Natural Language Processing Techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining*.

H. Yu & V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

J. Wiebe, E. Breck, C. Buckley, C. Cardie, P. Davis, B. Fraser, D. Litman, D. Pierce, E. Riloff & T. Wilson. 2002. NRRC Summer Workshop on Multiple-Perspective Question Answering: Final Report.

J. Wiebe & E. Riloff. 2005. Creating subjective and objective sentence classifi ers from unannotated texts. *Sixth International Conference on Intelligent Text Processing and Computational Linguistics*.

J. Wiebe, T. Wilson & C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 1(2).

Disambiguating Toponyms in News

Eric Garbin
Department of Linguistics
Georgetown University
Washington, DC 20057, USA
egarbin@cox.net

Inderjeet Mani
Department of Linguistics
Georgetown University
Washington, DC 20057, USA
im5@georgetown.edu

Abstract

This research is aimed at the problem of disambiguating toponyms (place names) in terms of a classification derived by merging information from two publicly available gazetteers. To establish the difficulty of the problem, we measured the degree of ambiguity, with respect to a gazetteer, for toponyms in news. We found that 67.82% of the toponyms found in a corpus that were ambiguous in a gazetteer lacked a local discriminator in the text. Given the scarcity of human-annotated data, our method used unsupervised machine learning to develop disambiguation rules. Toponyms were automatically tagged with information about them found in a gazetteer. A toponym that was ambiguous in the gazetteer was automatically disambiguated based on preference heuristics. This automatically tagged data was used to train a machine learner, which disambiguated toponyms in a human-annotated news corpus at 78.5% accuracy.

1 Introduction

Place names, or toponyms, are ubiquitous in natural language texts. In many applications, including Geographic Information Systems (GIS), it is necessary to interpret a given toponym mention as a particular entity in a geographical database or gazetteer. Thus the mention "Washington" in "He visited Washington last year" will need to be interpreted as a reference to either the city Washington, DC or the U.S. state of Washington, and "Berlin" in "Berlin is cold in the winter" could mean Berlin, New Hampshire or Berlin, Germany, among other possibilities. While there has been a considerable body of work distinguishing between a toponym and other kinds of names (e.g., person names), there has been relatively little work on resolving which place and what kind of place given a classification of kinds of places in a gazetteer. Disambiguated toponyms can be used in a GIS to highlight a position on a map corresponding to the coordinates of the place, or to draw a polygon representing the boundary.

In this paper, we describe a corpus-based method for disambiguating toponyms. To establish the difficulty of the problem, we began by quantifying the degree of ambiguity of toponyms in a corpus with respect to a U.S. gazetteer. We then carried out a corpus-based investigation of features that could help disambiguate toponyms. Given the scarcity of human-annotated data, our method used unsupervised machine learning to develop disambiguation rules. Toponyms were automatically tagged with information about them found in a gazetteer. A toponym that was ambiguous in the gazetteer was automatically disambiguated based on preference heuristics. This automatically tagged data was used to train the machine learner. We compared this method with a supervised machine learning approach trained on a corpus annotated and disambiguated by hand.

Our investigation targeted toponyms that name cities, towns, counties, states, countries or national capitals. We sought to classify each toponym as a *national capital*, a *civil political/administrative region*, or a *populated place* (administration unspecified). In the vector model of GIS, the type of place crucially determines the geometry chosen to represent it (e.g., point, line or polygon) as well as any reasoning about geographical inclusion. The class of the toponym can be useful in "grounding" the toponym to latitude and longitude coordinates,

Entry Number	Toponym	U.S. County	U.S. State	Lat-Long (dddmmss)	Elevation (ft. above sea level)	Class
110	Acton	Middlesex	Massachusetts	422906N-0712600W	260	Ppl (populated place)
111	Acton	Yellowstone	Montana	455550N-1084048W	3816	Ppl
112	Acton	Los Angeles	California	342812N-1181145W	2720	Ppl

Table 1. Example GNIS entries for an ambiguous toponym

but it can also go beyond grounding to support spatial reasoning. For example, if the province is merely grounded as a point in the data model (e.g., if the gazetteer states that the centroid of a province is located at a particular latitude-longitude) then without the class information, the inclusion of a city within a province can't be established. Also, resolving multiple cities or a unique capital to a political region mentioned in the text can be a useful adjunct to a map that lacks political boundaries or whose boundaries are dated.

It is worth noting that our classification is more fine-grained than efforts like the EDT task in Automatic Content Extraction[1] program (Mitchell and Strassel 2002), which distinguishes between toponyms that are a Facility "Alfredo Kraus Auditorium", a Location "the Hudson River", and Geo-Political Entities that include territories "U.S. heartland", and metonymic or other derivative place references "Russians", "China (offered)", "the U.S. company", etc. Our classification, being gazetteer based, is more suited to GIS-based applications.

2 Quantifying Toponym Ambiguity

2.1 Data

We used a month's worth of articles from the New York Times (September 2001), part of the English Gigaword (LDC 2003). This corpus consisted of 7,739 documents and, after SGML stripping, 6.51 million word tokens with a total size of 36.4MB). We tagged the corpus using a list of place names from the USGS Concise Gazetteer (GNIS). The resulting corpus is called MAC1, for "Machine Annotated Corpus 1". GNIS covers cities, states, and counties in the U.S., which are classified as "civil" and "populated place" *geographical entities.* A *geographical entity* is an entity on the Earth's surface that can be represented by some geometric specification in a GIS; for example, as a point, line or polygon. GNIS also covers 53 other types of geo-entities, e.g., "valley," "summit", "water" and "park." GNIS has 37,479 entries, with 27,649 distinct toponyms, of which 13,860 toponyms had multiple entries in the GNIS (i.e., were ambiguous according to GNIS). Table 1 shows the entries in GNIS for an ambiguous toponym.

2.2 Analysis

Let E be a set of elements, and let F be a set of features. We define a feature g in F to be a *disambiguator* for E iff for all pairs $<e_x, e_y>$ in E X E, $g(e_x) \neq g(e_y)$ and neither $g(e_x)$ nor $g(e_y)$ are null-valued. As an example, consider the GNIS gazetteer in Table 1, let F = {U.S. County, U.S. State, Lat-Long, and Elevation}. We can see that each feature in F is a disambiguator for the set of entries E = {110, 111, 112}.

Let us now characterize the mapping between texts and gazetteers. A string s1 in a text is said to be a *discriminator within a window w* for another string s2 no more than w words away if s1 matches a disambiguator d for s2 in a gazetteer. For example, "MT" is a discriminator within a window 5 for the toponym "Acton" in "Acton, MT," since "MT" occurs within a ±5-word window of "Acton" and matches, via an abbreviation, "Montana", the value of a GNIS disambiguator U.S. State (here the tokenized words are "Acton", ",", and "MT").

A trie-based lexical lookup tool (called LexScan) was used to match each toponym in GNIS against the corpus MAC1. Of the 27,649 distinct toponyms

[1] www.ldc.upenn.edu/Projects/ACE/

in GNIS, only 4553 were found in the corpus (note that GNIS has only U.S. toponyms). Of the 4553 toponyms, 2911 (63.94%) were “bare” toponyms, lacking a local discriminator within a ±5-word window that could resolve the name.

Of the 13,860 toponyms that were *ambiguous* according to GNIS, 1827 of them were found in MAC1, of which only 588 had discriminators within a ±5-word window (i.e., discriminators which matched gazetteer features that disambiguated the toponym). Thus, **67.82%** of the 1827 toponyms found in MAC1 that were ambiguous in GNIS lacked a discriminator.

This 67.82% proportion is only an estimate of true toponym ambiguity, even for the sample MAC1. There are several sources of error in this estimate: (i) World cities, capitals and countries were not yet considered, since GNIS only covered U.S. toponyms. (ii) In general, a single feature (e.g., County, or State) may not be sufficient to disambiguate a set of entries. It is of course possible for two different places named by a common toponym to be located in the same county in the same state. However, there were no toponyms with this property in GNIS. (iii) A string in MAC1 tagged by GNIS lexical lookup as a toponym may not have been a place name at all (e.g., “Lord Acton lived …”). Of the toponyms that were spurious, most were judged by us to be common words and person names. This should not be surprising, as 5341 toponyms in GNIS are also person names according to the U.S. Census Bureau[2] (iv) LexScan wasn’t perfect, for the following reasons. First, it sought only exact matches. Second, the matching relied on expansion of standard abbreviations. Due to non-standard abbreviations, the number of true U.S. toponyms in the corpus likely exceeded 4553. Third, the matches were all case-sensitive: while case-insensitivity caused numerous spurious matches, case-sensitivity missed a more predictable set, i.e. all-caps dateline toponyms or lower-case toponyms in Internet addresses.

Note that the 67.82% proportion is just an estimate of *local* ambiguity. Of course, there are often non-local discriminators (outside the ±5-word windows); for example, an initial place name reference could have a local discriminator, with subsequent references in the article lacking local discriminators while being *coreferential* with the initial reference. To estimate this, we selected cases where a toponym was discriminated on its first mention. In those cases, we counted the number of times the toponym was repeated in the same document without the discriminator. We found that **73%** of the repetitions lacked a local discriminator, suggesting an important role for coreference (see Sections 4 and 5).

3 Knowledge Sources for Automatic Disambiguation

To prepare a toponym disambiguator, we required a gazetteer as well as corpora for training and testing it.

3.1 Gazetteer

To obtain a gazetteer that covered worldwide information, we harvested countries, country capitals, and populous world cities from two websites ATLAS[3] and GAZ[4], to form a consolidated gazetteer (WAG) with four features G1,..,G4 based on geographical inclusion, and three classes, as shown in Table 2. As an example, an entry for Aberdeen could be the following feature vector: G1=United States, G2=Maryland, G3=Harford County, G4=Aberdeen, CLASS=ppl.

We now briefly discuss the merging of ATLAS and GAZ to produce WAG. ATLAS provided a simple list of countries and their capitals. GAZ recorded the country as well as the population of 700 cities of at least 500,000 people. If a city was in both sources, we allowed two entries but ordered them in WAG to make the more specific type (e.g. “capital”) the default sense, the one that LexScan would use. Accents and diacritics were stripped from WAG toponyms by hand, and aliases were associated with standard forms. Finally, we merged GNIS state names with these, as well as abbreviations discovered by our abbreviation expander.

3.2 Corpora

We selected a corpus consisting of 15,587 articles from the complete Gigaword Agence France

[2] www.census.gov/genealogy/www/freqnames.html

[3] . www.worldatlas.com

[4] www.worldgazetteer.com

Presse, May 2002. LexScan was used to tag, insensitive to case, all WAG toponyms found in this corpus, with the attributes in Table 2. If there were multiple entries in WAG for a toponym, LexScan only tagged the preferred sense, discussed below. The resulting tagged corpus, called MAC-DEV,

Tag Attribute	Description
CLASS	**Civil** (Political Region or Administrative Area, e.g. Country, Province, County), **Ppl** (Populated Place, e.g. City, Town), **Cap** (Country Capital, Provincial Capital, or County Seat)
G1	Country
G2	Province (State) or Country-Capital
G3	County or Independent City
G4	City, Town (Within County)

Table 2: WAG Gazetteer Attributes

Corpus	Size	Use	How Annotated
MAC1	6.51 million words with 61,720 place names (4553 distinct) from GNIS	Ambiguity Study (Gigaword NYT Sept. 2001) (Section 2)	LexScan of all senses, no attributes marked
MAC-DEV	5.47 million words with 124,175 place names (1229 distinct) from WAG	Development Corpus (Gigaword AFP May 2002) (Section 4)	LexScan using attributes from WAG, with heuristic preference
MAC-ML	6.21 million words with 181,866 place names (1322 distinct) from WAG	Machine Learning Corpus (Gigaword AP Worldwide January 2002) (Section 5)	LexScan using attributes from WAG, with heuristic preference
HAC	83,872 words with 1275 place names (435 distinct) from WAG.	Human Annotated Corpus (from TimeBank 1.2, and Gigaword NYT Sept. 2001 and June 2002) (Section 5)	LexScan using WAG, with attributes and sense being manually corrected

Table 3. Summary of Corpora

Term found with *Cap*	T-test *Civil*	T-test *Ppl*	Term found with *Ppl*	T-test *Civil*	T-test *Cap*	Term found with *Civil*	T-test *Ppl*	T-test *Cap*
'stock'	4	4	'winter'	3.61	3.61	'air'	3.16	3.16
'exchange'	4.24	4.24	'telephone'	3.16	3.16	'base'	3.16	3.16
'embassy'	3.61	3.61	'port'	3.46	3.46	'accuses'	3.61	3.61
'capital'	**1.4**	**2.2**	'midfielder'	3.46	3.46	'northern'	5.57	5.57
'airport'	3.32	3.32	'city'	1.19	1.19	'airlines'	4.8	4.8
'summit'	4	4	**'near'**	**2.77**	**3.83**	'invaded'	3.32	3.32
'lower'	3.16	3.16	'times'	3.16	3.16	**'southern'**	**3.87**	**6.71**
'visit'	**4.61**	**4.69**	'southern'	3.87	3.87	'friendly'	4	4
'conference'	4.24	4.24	**'yen'**	**4**	**0.56**	'state-run'	3.32	3.32
'agreement'	3.16	3.16	**'attack'**	**0.18**	**3.87**	'border'	7.48	7.48

Table 4. Top 10 terms disambiguating toponym classes

was used as a development corpus for feature exploration. To disambiguate the sense for a toponym that was ambiguous in WAG, we used two preference heuristics. First, we searched

MAC1 for two dozen highly frequent ambiguous toponym strings (e.g., "Washington", etc.), and observed by inspection which sense predominated in MAC1, preferring the predominant sense for each of these frequently mentioned toponyms. For example, in MAC1, "Washington" was predominantly a Capital. Second, for toponyms outside this most frequent set, we used the following specificity-based preference: *Cap.* > *Ppl* > *Civil*. In other words, we prefer the more specific sense; since there are a smaller number of Capitals than Populated places, we prefer Capitals to Populated Places.

For machine learning, we used the Gigaword Associated Press Worldwide January 2002 (15,999 articles), tagged in the same way by LexScan as MAC-DEV was. This set was called MAC-ML. Thus, MAC1, MAC-DEV, and MAC-ML were all generated automatically, without human supervision.

For a blind test corpus with human annotation, we opportunistically sampled three corpora: MAC1, TimeBank 1.2[5] and the June 2002 New York Times from the English Gigaword, with the first author tagging a random 28, 88, and 49 documents respectively from each. Each tag in the resulting human annotated corpus (HAC) had the WAG attributes from Table 2 with manual correction of all the WAG attributes. A summary of the corpora, their source, and annotation status is shown in Table 3.

4 Feature Exploration

We used the tagged toponyms in MAC-DEV to explore useful features for disambiguating the classes of toponyms. We identified single-word terms that co-occurred significantly with classes within a k-word window (we tried k= ±3, and k=±20). These terms were scored for pointwise mutual information (MI) with the classes. Terms with average tf.idf of less than 4 in the collection were filtered out as these tended to be personal pronouns, articles and prepositions.

To identify which terms helped select for particular classes of toponyms, the set of 48 terms whose MI scores were above a threshold (-11, chosen by inspection) were filtered using the student's t-statistic, based on an idea in (Church and Hanks 1991). The t-statistic was used to compare the distribution of the term with one class of toponym to its distribution with other classes to assess whether the underlying distributions were significantly different with at least 95% confidence. The results are shown in Table 4, where scores for a term that occurred jointly in a window with at least one other class label are shown in bold. A t-score > 1.645 is a significant difference with 95% confidence. However, because joint evidence was scarce, we eventually chose not to eliminate Table 4 terms such as 'city' (t =1.19) as features for machine learning. Some of the terms were significant disambiguators between only one pair of classes, e.g. 'yen,' 'attack,' and 'capital,' but we kept them on that basis.

Feature Name	**Description**
Abbrev	Value is *true* iff the toponym is abbreviated.
AllCaps	Value is *true* iff the toponym is all capital letters.
Left/Right Pos{1,.., k}	Values are the ordered tokens up to *k* positions to the left/right
WkContext	Value is the *set* of MI collocated terms found in windows of ± *k* tokens (to the left and right)
TagDiscourse	Value is the *set* of CLASS values represented by all toponyms from the document: e.g., the set {*civil, capital, ppl*}
CorefClass	Value is the CLASS if any for a prior mention of a toponym in the document, or *none*

Table 5. Features for Machine Learning

Based on the discovered terms in experiments with different window sizes, and an examination of MAC1 and MAC-DEV, we identified a final set of features that, it seemed, might be useful for machine learning experiments. These are shown in Table 5. The features *Abbrev* and *Allcaps* describe evidence internal to the toponym:

[5] www.timeml.org

an abbreviation may indicate a state (Mass.), territory (N.S.W.), country (U.K.), or some other *civil* place; an all-caps toponym might be a *capital* or *ppl* in a dateline. The feature sets *LeftPos* and *RightPos* target the ±*k* positions in each window as ordered tokens, but note that only windows with a MI term are considered. The domain of *WkContext* is the window of ±k tokens around a toponym that contains a MI collocated term.

We now turn to the global discourse-level features. The domain for *TagDiscourse* is the whole document, which is evaluated for the set of toponym classes present: this information may reflect the discourse topic, e.g. a discussion of U.S. sports teams will favor mentions of cities over states or capitals. The feature *CorefClass* implements a one sense per discourse strategy, motivated by our earlier observation (from Section 2) that 73% of subsequent mentions of a toponym that was discriminated on first mention were expressed without a local discriminator.

5 Machine Learning

The features in Table 5 were used to code feature vectors for a statistical classifier. The results are shown in Table 6. As an example, when the Ripper classifier (Cohen 1996) was trained on MAC-ML with a window of *k*= ±3 word tokens, the predictive accuracy when tested using cross-validation MAC-ML was 88.39% ±0.24 (where 0.24 is the standard deviation across 10 folds).

		Accuracy on Test Set			
		Window = ±3		**Window = ±20**	
Training Set	**Test Set**	**Predictive Accuracy**	**Recall, Precision, F-measure**	**Predictive Accuracy**	**Recall, Precision, F-measure**
MAC-ML	MAC-ML *(cross-validation)*	**88.39** ± 0.24 (*Civ.* 65.0)	*Cap* r70 p88 f78 *Civ.* r94 p90 f92 *Ppl* r87 p82 f84 Avg. r84 p87 **f85**	80.97 ± 0.33 (*Civ.* 57.1)	*Cap* r61 p77 f68 *Civ.* r83 p86 f84 *Ppl* r81 p72 f76 Avg. r75 p78 f76
MAC-DEV	MAC-DEV *(cross-validation)*	87.08 ± 0.28 (*Civ.* 57.8)	*Cap* r74 p87 f80 *Civ.* r93 p88 f91 *Ppl* r82 p80 f81 Avg. r83 p85 f84	81.36 ± 0.59 (*Civ.* 59.3)	*Cap* r49 p78 f60 *Civ.* r92 p81 f86 *Ppl* r56 p70 f59 Avg. r66 p77 f68
MAC-DEV	HAC	68.66 (*Civ.* 59.7)	*Cap* r50 p71 f59 *Civ.* r93 p70 f80 *Ppl* r24 p57 f33 Avg. r56 p66 f57	65.33 (*Civ.* 50.7)	*Cap* r100 p100 f100 *Civ.* r84 p62 f71 *Ppl* r43 p71 f54 Avg. r76 p78 f75
HAC	HAC *(cross-validation)*	77.5 ± 2.94 (*Ppl* 72.9)	*Cap* r70 p97 f68 *Civ.* r34 p94 f49 *Ppl* r98 p64 f77 Avg. r67 p85 f65	73.12 ± 3.09 (*Ppl* 51.3)	*Cap* r17 p90 f20 *Civ.* r63 p76 f68 *Ppl* r84 p73 f77 Avg. r54 p79 f55
MAC-DEV+MAC-ML	MAC-DEV+MAC-ML *(cross-validation)*	86.76 ± 0.18 (*Civ.* 60.7)	*Cap* r70 p89 f78 *Civ.* r94 r88 f91 *Ppl* r81 p80 f80 Avg. r82 p86 f83	79.70 ± 0.30 (*Civ.* 59.7)	*Cap* r56 p73 f63 *Civ.* r83 p86 f84 *Ppl* r80 p68 f73 Avg. r73 p76 f73
MAC-DEV+MAC-ML	HAC	73.07 (*Civ.* 51.7)	*Cap* r71 p83 f77 *Civ.* r91 p69 f79 *Ppl* r45 f81 f58 Avg. r69 p78 f71	**78.30** (*Civ.* 50)	*Cap* r100 p63 f77 *Civ.* r91 p75 f82 *Ppl* r63 p88 f73 Avg. r85 p75 **f77**

Table 6. Machine Learning Accuracy

The majority class (**Civil**) had the predictive accuracy shown in parentheses. (When tested on a different set from the training set, cross-validation wasn't used). Ripper reports a confusion matrix for each class; Recall, Precision, and F-measure for these classes are shown, along with their average across classes.

In all cases, Ripper is significantly better in predictive accuracy than the majority class. When testing using cross-validation on the same machine-annotated corpus as the classifier was trained on, performance is comparable across corpora, and is in the high 80%, e.g., **88.39** on MAC-ML (k=±3). Performance drops substantially when we train on machine-annotated corpora but test on the human-annotated corpus (HAC) (the unsupervised approach), or when we both train and test on HAC (the supervised approach). The noise in the auto-generated classes in the machine-annotated corpus is a likely cause for the lower accuracy of the unsupervised approach. The poor performance of the supervised approach can be attributed to the lack of human-annotated training data: HAC is a small, 83,872-word corpus.

Rule Description (Window = ±3)	**Coverage of Examples in Testing (Accuracy)**
If not AllCaps(P) and Right-Pos1(P,'SINGLE_QUOTE') and Civil ∈ TagDiscourse Then Civil(P).	5/67 (100%)
If not AllCaps(P) and Left-Pos1(P, *southern*) and Civil ∈ TagDiscourse Then Civil(P).	13/67 (100%)

Table 7. Sample Rules Learnt by Ripper

TagDiscourse was a critical feature; ignoring it during learning dropped the accuracy nearly 9 percentage points. This indicates that prior mention of a class increases the likelihood of that class. (Note that when inducing a rule involving a set-valued feature, Ripper tests whether an element is a member of that set-valued feature, selecting the test that maximizes information gain for a set of examples.) Increasing the *window size* only lowered accuracy when tested on the same corpus (using cross-validation); for example, an increase from ±3 words to ±20 words (intervening sizes are not shown for reasons of space) lowered the PA by 5.7 percentage points on MAC-DEV. However, increasing the *training set size* was effective, and this increase was more substantial for larger window sizes: combining MAC-ML with MAC-DEV improved accuracy on HAC by about 4.5% for *k*= ±3, but an increase of 13% was seen for *k* =±20. In addition, F-measure for the classes was steady or increased. As Table 6 shows, this was largely due to the increase in recall on the non-majority classes. The best performance when training Ripper on the machine-annotated MAC-DEV+MAC-ML and testing on the human-annotated corpus HAC was 78.30.

Another learner we tried, the SMO support-vector machine from WEKA (Witten and Frank 2005), was marginally better, showing **81.0** predictive accuracy training and testing on MAC-DEV+MAC-ML (ten-fold cross-validation, k=±20) and **78.5** predictive accuracy training on MAC-DEV+MAC-ML and testing on HAC (k=±20). Ripper rules are of course more transparent: example rules learned from MAC-DEV are shown in Table 7, along with their coverage of feature vectors and accuracy on the test set HAC.

6 Related Work

Work related to toponym tagging has included harvesting of gazetteers from the Web (Uryupina 2003), hand-coded rules to place name disambiguation, e.g., (Li et al. 2003) (Zong et al. 2005), and machine learning approaches to the problem, e.g., (Smith and Mann 2003). There has of course been a large amount of work on the more general problem of word-sense disambiguation, e.g., (Yarowsky 1995) (Kilgarriff and Edmonds 2002). We discuss the most relevant work here.

While (Uryupina 2003) uses machine learning to induce gazetteers from the Internet, we merely download and merge information from two popular Web gazetteers. (Li et al. 2003) use a statistical approach to tag place names as a LOCation class. They then use a heuristic approach to location normalization, based on a combination of hand-coded pattern-matching rules as well as discourse features based on co-occurring toponyms (e.g., a document with "Buffalo", "Albany" and "Rochester" will likely have those toponyms disambiguated to New York state). Our *TagDiscourse* feature is more coarse-grained. Finally, they assume one sense per discourse in their rules, whereas we use it as a feature *CorefClass* for use in learning. Overall, our approach is based on unsupervised machine learning, rather than hand-coded rules for location normalization.

(Smith and Mann 2003) use a "minimally supervised" method that exploits as training data toponyms that are found locally disambiguated, e.g., "Nashville, Tenn."; their disambiguation task is to identify the state or country associated with the toponym in test data that has those disambiguators stripped off. Although they report 87.38% accuracy on news, they address an easier problem than ours, since: (i) our earlier local ambiguity estimate suggests that as many as two-thirds of the gazetteer-ambiguous toponyms may be excluded from their test on news, as they would lack local discriminators (ii) the classes our tagger uses (Table 3) are more fine-grained. Finally, they use one sense per discourse as a bootstrapping strategy to expand the machine-annotated data, whereas in our case *CorefClas*s is used as a feature.

Our approach is distinct from other work in that it firstly, attempts to quantify toponym ambiguity, and secondly, it uses an unsupervised approach based on learning from noisy machine-annotated corpora using publicly available gazetteers.

7 Conclusion

This research provides a measure of the degree of of ambiguity with respect to a gazetteer for toponyms in news. It has developed a toponym disambiguator that, when trained on entirely machine annotated corpora that avail of easily available Internet gazetteers, disambiguates toponyms in a human-annotated corpus at 78.5% accuracy.

Our current project includes integrating our disambiguator with other gazetteers and with a geovisualization system. We will also study the effect of other window sizes and the combination of this unsupervised approach with minimally-supervised approaches such as (Brill 1995) (Smith and Mann 2003). To help mitigate against data sparseness, we will cluster terms based on stemming and semantic similarity.

The resources and tools developed here may be obtained freely by contacting the authors.

References

Eric Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. *ACL Third Workshop on Very Large Corpora*, Somerset, NJ, p. 1-13.

Ken Church, Patrick Hanks, Don Hindle, and William Gale. 1991. Using Statistics in Lexical Analysis. In U. Zernik (ed), *Lexical Acquisition: Using On-line Resources to Build a Lex*icon, Erlbaum, p. 115-164.

William Cohen. 1996. Learning Trees and Rules with Set-valued Features. Proceedings of AAAI 1996, Portland, Oregon, p. 709-716.

Adam Kilgarriff and Philip Edmonds. 2002. Introduction to the Special Issue on Evaluating Word Sense Disambiguation Systems. *Journal of Natural Language Engineering* 8 (4).

Huifeng Li, Rohini K. Srihari, Cheng Niu, and Wei Li. 2003. A hybrid approach to geographical references in information extraction. *HLT-NAACL 2003 Workshop: Analysis of Geographic References*, Edmonton, Alberta, Canada.

LDC. 2003. Linguistic Data Consortium: English Gigaword www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05

Alexis Mitchell and Stephanie Strassel. 2002. Corpus Development for the ACE (Automatic Content Extraction) Program. Linguistic Data Consortium www.ldc.upenn.edu/Projects/LDC_Institute/Mitchell/ACE_LDC_06272002.ppt

David Smith and Gideon Mann. 2003. Bootstrapping toponym classifiers. *HLT-NAACL 2003 Workshop: Analysis of Geographic References*, p. 45-49, Edmonton, Alberta, Canada.

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

Olga Uryupina. 2003. Semi-supervised learning of geographical gazetteers from the internet. *HLT-NAACL 2003 Workshop: Analysis of Geographic References*, Edmonton, Alberta, Canada.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. Proceedings of *ACL 1995*, Cambridge, Massachusetts.

Wenbo Zong, Dan Wu, Aixin Sun, Ee-Peng Lim, and Dion H. Goh. 2005. On Assigning Place Names to Geography Related Web Pages. *Joint Conference on Digital Libraries (JCDL2005)*, Denver, Colorado.

A Semantic Approach to Recognizing Textual Entailment

Marta Tatu and **Dan Moldovan**
Language Computer Corporation
Richardson, TX 75080, USA
marta,moldovan@languagecomputer.com

Abstract

Exhaustive extraction of semantic information from text is one of the formidable goals of state-of-the-art NLP systems. In this paper, we take a step closer to this objective. We combine the semantic information provided by different resources and extract new semantic knowledge to improve the performance of a *recognizing textual entailment* system.

1 Recognizing Textual Entailment

While communicating, humans use different expressions to convey the same meaning. Therefore, numerous NLP applications, such as, Question Answering, Information Extraction, or Summarization require computational models of language that recognize if two texts semantically overlap. Trying to capture the major inferences needed to understand equivalent semantic expressions, the PASCAL Network proposed the *Recognizing Textual Entailment* (RTE) challenge (Dagan et al., 2005). Given two text fragments, the task is to determine if the meaning of one text (the entailed hypothesis, H) can be inferred from the meaning of the other text (the entailing text, T).

Given the wide applicability of this task, there is an increased interest in creating systems which detect the semantic entailment between two texts. The systems that participated in the Pascal RTE challenge competition exploit various inference elements which, later, they combine within statistical models, scoring methods, or machine learning frameworks. Several systems (Bos and Markert, 2005; Herrera et al., 2005; Jijkoun and de Rijke, 2005; Kouylekov and Magnini, 2005; Newman et al., 2005) measured the word overlap between the two text strings. Using either statistical or WordNet's relations, almost all systems considered lexical relationships that indicate entailment. The degree of similarity between the syntactic parse trees of the two texts was also used as a clue for entailment by several systems (Herrera et al., 2005; Kouylekov and Magnini, 2005; de Salvo Braz et al., 2005; Raina et al., 2005). Several groups used logic provers to show the entailment between T and H (Bayer et al., 2005; Bos and Markert, 2005; Fowler et al., 2005; Raina et al., 2005) and some of them made use of world knowledge axioms to increase the logic prover's power of inference (Bayer et al., 2005; Bos and Markert, 2005; Fowler et al., 2005).

In this paper, we describe a novel technique which employs a set of *semantic axioms* in its attempt to exhaustively extract semantic knowledge from texts. In order to show the contribution that our semantic information extraction method brings, we append it as an additional module to an already existing system that participated in the RTE challenge. Our system (Fowler et al., 2005), first, transforms the text T and the hypothesis H into semantically enhanced logic forms, and, then, the integrated logic prover tries to prove or disprove the entailment using a set of world-knowledge axioms (*die of blood loss* $\rightarrow$ *bleed to death*), linguistic rewriting rules which break down complex syntactic structures, like coordinating conjunctions, and WordNet-based lexical chains axioms (*buy/VB/1* $\rightarrow$ *pay/VB/1*).

2 Approach

We believe that a logic-based semantic approach is highly appropriate for the RTE task[1]. Text T semantically entails H if its meaning logically implies the meaning of H. Because the set of semantic relations encoded in a text represents its meaning, we need to identify all the semantic relations that hold between the constituents of T and, subsequently, between the constituents of H to understand the meaning of each text. It should be noted that state-of-the-art semantic parsers extract only some of the semantic relations encoded in a given text. To complete this information, we need semantic axioms that augment the extracted knowledge and, thus, provide a better coverage of the text's semantics. Once we gather this information, we state that text T entails hypothesis H if and only if we find similar relations between a concept from T and a semantically analogous concept from H. By analogous concepts, we mean identical concepts, or words connected by a chain of SYNONYMY, HYPERNYMY or morphological derivation relations in WordNet.

Because the set of semantic elements identified by a semantic parser does not necessarily convey the complete meaning of a sentence, we shall use a set of semantic axioms to infer the missing pieces of information. By combining two semantic relations or by using the FrameNet's frame elements identified in a given text, we derive new semantic information.

In order to show if T entails H, we analyze their meanings. Our approach to semantic entailment involves the following stages:

1. We convert each text into logic form (Moldovan and Rus, 2001). This conversion includes part-of-speech tagging, parse tree generation, and name entity recognition.

2. Using our semantic parser, we identify some of the semantic relations encoded in the analyzed texts. We note that state-of-the-art semantic parsers cannot discover all the semantic relations conveyed implicitly or explicitly by the text. This problem compromises our system's performance. To obtain the complete set of semantic relations that represents the meaning of the given texts, we introduce a new step in our algorithm.

3. We add *semantic axioms* to the already created set of world knowledge, NLP, and WordNet-based lexical chain (Moldovan and Novischi, 2002) axioms that assist the logic prover in its search for proofs. We developed semantic axioms that show how two semantic relations can be combined. This will allow the logic prover to combine, whenever possible, semantic instances in order to infer new semantic relationships. The instances of relations that participate in semantic combinations can be either provided by the text or annotated between WordNet synsets. We also exploit other sources of semantic information from the text. For example, the frames encoded in the text sentence provide information which complements the meaning given by the semantic relations. Our second type of axioms derive semantic relations between the frame elements of a given FrameNet frame.

We claim that the process of applying the semantic axioms, given the semantic relations detected by a semantic parser, will capture the complete semantic information expressed by a text fragment. In this paper, we show the usefulness of this procedure for the RTE task, but we are convinced that it can be used by any system which plans to extract the entire semantic information from a given text.

4. We load the COGEX logic prover (Moldovan et al., 2003) which operates by "reductio ad absurdum" with H's negated form and T's predicates. These clauses are weighted in the order in which they should be chosen to participate in the search. To ensure that H will be the last clause to participate, we assign it the largest value. The logic prover searches for new inferences that can be made using the smallest weight clauses. It also assigns a value to each inference based on the axiom it used to derive it. This process continues until the set of clauses is empty. If a refutation is found, the proof is complete. If a contradiction cannot be found, then the predicate arguments are relaxed and, if the argument relaxation fails, then predicates are dropped until a proof by refutation is found. Its score will be computed by deducting points for each argument relaxation and predicate removal. If this value falls below a threshold, then T does not entail H. Otherwise, the (T, H) pair is a true entailment.

We present a textual entailment example to show the steps of our approach. This proof will not

[1]After all, the *entailment, inference,* and *equivalence* terms originated from logic.

T	John and his son, George, emigrated with Mike, John's uncle, to US in 1969.
$\mathcal{LF}_T$	John$(x_1) \wedge$ son$(x_2) \wedge$ George$(x_3) \wedge \textsc{isa}(x_3, x_2) \wedge \textsc{kin}(x_1, x_3) \wedge$ emigrate$(e_1) \wedge \textsc{agt}(x_1, e_1) \wedge \textsc{agt}(x_2, e_1) \wedge$ Mike$(x_4) \wedge$ uncle$(x_5) \wedge \textsc{isa}(x_4, x_5) \wedge \textsc{kin}(x_1, x_4) \wedge$ US$(x_6) \wedge \textsc{loc}(e_1, x_6) \wedge$ 1969$(x_7) \wedge \textsc{tmp}(e_1, x_7)$.
$T_{Departing}$	[John and his son, George,]$_{Theme.fe}$ *emigrated* [with Mike, John's uncle,]$_{Cotheme.fe}$ to [US]$_{Goal.fe}$ in [1969]$_{Time.fe}$.
$T_{Kinship}$	[John]$_{Ego.fe}$ and his *son*, [George,]$_{Alter.fe}$ emigrated with [Mike]$_{Alter.fe}$, [John]$_{Ego.fe}$'s *uncle*, to US in 1969.
T_{Axiom_1}	$\textsc{kin}(w_1, w_2) \leftrightarrow \textsc{kin}(w_2, w_1)$ $\textsc{kin}(x_1, x_3) \rightarrow \textsc{kin}(x_3, x_1)$ $(\textsc{kin}(John, George) \rightarrow \textsc{kin}(George, John))$
T_{Axiom_2}	$\textsc{kin} \circ \textsc{kin} = \textsc{kin}$ $(\textsc{kin}(w_1, w_2) \wedge \textsc{kin}(w_2, w_3) \rightarrow \textsc{kin}(w_1, w_3))$ $\textsc{kin}(x_3, x_1) \wedge \textsc{kin}(x_1, x_4) \rightarrow \textsc{kin}(x_3, x_4)$ $(\textsc{kin}(George, Mike))$
T_{Axiom_3}	$\textsc{departing\_f} \rightarrow \textsc{loc}(Theme.fe, Goal.fe)$ $(\textsc{loc}(John, US) \wedge \textsc{loc}(George, US))$
T_{Axiom_4}	$\textsc{departing\_f} \rightarrow \textsc{loc}(Cotheme.fe, Goal.fe)$ $(\textsc{loc}(Mike, US))$
$T_{Semantics}$	$\textsc{kin}(John, George)$, $\textsc{kin}(John, Mike)$, $\textsc{kin}(George, Mike)$, $\textsc{loc}(John, US)$, $\textsc{loc}(George, US)$, $\textsc{loc}(Mike, US)$, $\textsc{tmp}(emigrate, 1969)$, $\textsc{agt}(John, emigrate)$, $\textsc{agt}(George, emigrate)$
H	George and his relative, Mike, came to America.
$\mathcal{LF}_H$	George$(x_1) \wedge$ relative$(x_2) \wedge$ Mike$(x_3) \wedge \textsc{isa}(x_3, x_2) \wedge \textsc{kin}(x_1, x_3) \wedge$ come$(e_1) \wedge \textsc{agt}(x_1, e_1) \wedge \textsc{agt}(x_2, e_1) \wedge$ America(x_4) $\wedge \textsc{loc}(e_1, x_2)$
$H_{Arriving}$	[George and his relative, Mike,]$_{Theme.fe}$ *came* to [America]$_{Goal.fe}$.
$H_{Kinship}$	[George]$_{Ego.fe}$ and his *relative*, [Mike]$_{Alter.fe}$, came to America.
H_{Axiom_1}	$\textsc{arriving\_f} \rightarrow \textsc{loc}(Theme.fe, Goal.fe)$ $(\textsc{loc}(George, America) \wedge \textsc{loc}(Mike, America))$
$H_{Semantics}$	$\textsc{kin}(George, Mike)$, $\textsc{loc}(George, America)$, $\textsc{loc}(Mike, America)$

Table 1: Entailment proof example. Table 2 lists the semantic relations and their abbreviations. Sections 3.2 and 4.1 will detail the semantics behind the axioms $T_{Axiom_1}, T_{Axiom_2}, T_{Axiom_3}, T_{Axiom_4}$, and H_{Axiom_1}.

make use of any world knowledge axioms. Let the text T be *John and his son, George, emigrated with Mike, John's uncle, to US in 1969* and the entailed hypothesis H *George and his relative, Mike, came to America*. Our system transforms each text into its corresponding semantically enhanced logic form ($\mathcal{LF}_T$ and $\mathcal{LF}_H$ in Table 1). Then, the logic prover uses the newly added semantic axioms to derive extra semantic information from T and H (for example, *George* and *Mike* are relatives, but T does not explicitly specify this), after another preprocessing step which identifies the frame elements of each frame encoded in the two texts ($T_{Departing}$, $T_{Kinship}$, $H_{Arriving}$, $H_{Kinship}$). In our example, the axioms T_{Axiom_1} and T_{Axiom_2} denote the *symmetry* and the *transitivity* of the KINSHIP relation. T_{Axiom_3}, T_{Axiom_4} and H_{Axiom_1} are the frame-related axioms used by the logic prover. The $T_{Semantics}$ and $H_{Semantics}$ rows (Table 1) summarize the meaning of T and H. We note that half of these semantic instances were extracted using the semantic axioms. Once the lexical chains between the concepts in T and the ones from H are computed, the entailment becomes straightforward. We represented, graphically, the meaning of the two texts in Figure 1. We also show the links between the analogous concepts that help prove the entailment.

In the coming sections of the paper, we detail the process of semantic axiom generation. We start with a summary of the axioms that combine two semantic relations.

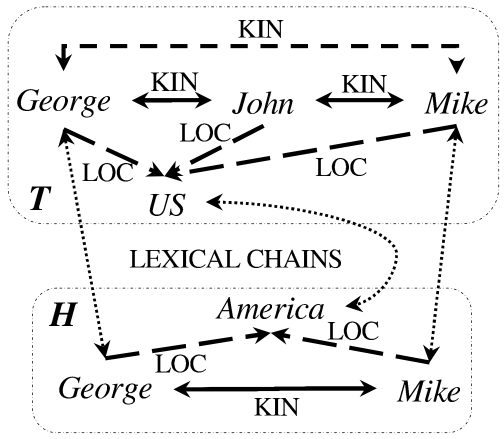


Figure 1: $T_{Semantics}$ and $H_{Semantics}$. The solid arrows represent the relations identified by the semantic parser. The dotted arrows symbolize the lexical chains between concepts in T and their analogous concepts in H (US_T and $America_H$ belong to the same WordNet synset). The dash arrows denote the relations inferred by combining two semantic relations. The long dash arrows indicate the relations between frame elements.

3 Semantic Calculus

3.1 Semantic relations

For this study, we adopt a revised version of the semantic relation set proposed by (Moldovan et al., 2004). Table 2 enumerates the semantic relations that we consider[2].

[2]See (Moldovan et al., 2004) for definitions and examples.

POSSESSION (POS)	MAKE-PRODUCE (MAK)	RECIPIENT (REC)	THEME-PATIENT (THM)
KINSHIP (KIN)	INSTRUMENT (INS)	FREQUENCY (FRQ)	RESULT (RSL)
PROPERTY-ATTRIBUTE (PAH)	LOCATION-SPACE (LOC)	INFLUENCE (IFL)	STIMULUS (STI)
AGENT (AGT)	PURPOSE (PRP)	ASSOCIATED WITH (OTH)	EXTENT (EXT)
TEMPORAL (TMP)	SOURCE-FROM (SRC)	MEASURE (MEA)	PREDICATE (PRD)
DEPICTION (DPC)	TOPIC (TPC)	SYNONYMY-NAME (SYN)	CAUSALITY (CSL)
PART-WHOLE (PW)	MANNER (MNR)	ANTONYMY (ANT)	JUSTIFICATION (JST)
HYPERNYMY (ISA)	MEANS (MNS)	PROBABILITY OF EXISTENCE (PRB)	GOAL (GOL)
ENTAIL (ENT)	ACCOMPANIMENT (ACC)	POSSIBILITY (PSB)	BELIEF (BLF)
CAUSE (CAU)	EXPERIENCER (EXP)	CERTAINTY (CRT)	MEANING (MNG)

Table 2: The set of semantic relations

3.2 Combinations of two semantic relations

Our goal is to devise semantic axioms for combinations of two relations, R_1 and R_2, by observing the semantic connection between the w_1 and w_3 words for which there exists at least one other word, w_2, such that $R_1(w_1, w_2)$ and $R_2(w_2, w_3)$ hold true[3].

Harabagiu and Moldovan (1998) tackled the problem of semantic combinations, for the first time. Their set of relations included the WordNet1.5 annotations and 12 relationships derived from the WordNet glosses[4]. In our research, unlike (Harabagiu and Moldovan, 1998), the semantic combinations use the relations identified in text with a rather minimal contribution from the WordNet relations.

Harabagiu and Moldovan (1998) also investigate the number of possible semantic combinations. Based on their properties, we can have up to eight combinations between any two semantic relations and their inverses, not counting the combinations between a semantic relation and itself[5]. For instance, given an asymmetric relation and a symmetric one which share the same part-of-speech for their arguments, we can produce four combinations. ISA $\circ$ ANT, ISA^{-1} $\circ$ ANT, ANT $\circ$ ISA, and ANT $\circ$ ISA^{-1} are the four possible distinct combinations between HYPERNYMY and ANTONYMY. "$\circ$" symbolizes the semantic composition between two relations compatible with respect to the part-of-speech of their arguments: for any two concepts, w_1 and w_3, $(R_i \circ R_j)(w_1, w_3)$ if and only if $\exists w_2$, a third concept, such that $R_i(w_1, w_2)$ and $R_j(w_2, w_3)$ hold. By R^{-1}, we denote the inverse of relation R: if $R(x, y)$, then $R^{-1}(y, x)$.

While analyzing the combinations, we observed some regularities within the semantic composition process. For example, $R_1^{-1} \circ R_2^{-1} = (R_2 \circ R_1)^{-1}$ for any, not necessarily distinct, semantic relations R_1 and R_2[6]. If one of the relations is symmetric ($R^{-1} = R$), the statement is still valid. Using $(R^{-1})^{-1} = R$ and the previous equality, we can reduce by half the number of semantic combinations that we have to compute for $R_1 \neq R_2$.

We plan to create a 40×40 matrix with all the possible combinations between any two semantic relations from the set we consider. Theoretically, we can have up to 27,556 semantic combinations, but only 25.79% of them are possible[7] (for example, MNR(r, v) and SYN(n, n) cannot be combined). Many combinations are not semantically significant either because they are very rare, like, KIN(n, n) $\circ$ TMP(n, v), or because they do not result into one of the 40 relations, for instance, PAH(a, n) $\circ$ AGT(n, v)[8]. We identified two approaches to the problem mentioned above. The first tries to fill one matrix cell at a time in a consecutive manner. The second approach tries to solve the semantic combinations we come upon in text corpora. As a result, we analyzed the RTE development corpus and we devised rules for some of the $R_i \circ R_j$ combinations that we encountered. We validated these axioms by man-

[3] $R(x, y)$ indicates that relation R holds between x and y.

[4] This set includes the AGENT, OBJECT, INSTRUMENT, BENEFICIARY, PURPOSE, ATTRIBUTE, REASON, STATE, LOCATION, THEME, TIME, and MANNER relations.

[5] Harabagiu and Moldovan (1998) lists the exact number of possible combinations for several WordNet relations and part-of-speech classes.

[6] The equality holds only if the two composition terms exist.

[7] On average, each semantic relation has 2.075 pairs of arguments. For example, SRC can connect two nouns (*US investor*), or an adjective and a noun (*American investor*) and, depending on its arguments, SRC will participate in different combinations. Out of the 27,556 combinations, only 7,109 are syntactically possible.

[8] n, v, a, and r stand for *noun*, *verb*, *adjective*, and *adverb*, respectively. As an example, $R(n, n)$ means that relation R can connect two nouns.

LOCATION ∘ PART-WHOLE = LOCATION (LOCATION(x, l_1) $\wedge$ PART-WHOLE(l_1, l_2) $\rightarrow$ LOCATION(x, l_2))
Example: John lives in Dallas, Texas.
LOCATION$(John, Dallas)$ and PART-WHOLE$(Dallas, Texas)$ imply that LOCATION$(John, Texas)$.
ISA ∘ ATTRIBUTE = ATTRIBUTE (ISA(x, y) $\wedge$ ATTRIBUTE(y, a) $\rightarrow$ ATTRIBUTE(x, a))
Example: Mike is a rich man.
If ISA$(Mike, man)$ and ATTRIBUTE$(man, rich)$, then ATTRIBUTE$(Mike, rich)$.
Similar statements can be made for other "attributes": LOCATION, TIME, SOURCE, etc.
ISA ∘ LOCATION = LOCATION (ISA(x, y) $\wedge$ LOCATION(y, l) $\rightarrow$ LOCATION(x, l))
Example: The man in the car, George, is an old friend of mine.
ISA$(George, man)$ and LOCATION(man, car) $\rightarrow$ LOCATION$(George, car)$
KINSHIP ∘ KINSHIP = KINSHIP (KINSHIP(x, y) $\wedge$ KINSHIP(y, z) $\rightarrow$ KINSHIP(x, z))
See example in Section 2.
THEME ∘ ISA^{-1} = THEME (THEME(e, y) $\wedge$ ISA(x, y) $\rightarrow$ THEME(e, x))
Example: Yesterday, John ate some fruits: an apple and two oranges.
THEME$(eat, fruit)$ $\wedge$ ISA$(apple, fruit)$ $\rightarrow$ THEME$(eat, apple)$
THEME ∘ PART-WHOLE^{-1} = THEME (THEME(e, y) $\wedge$ PART-WHOLE(x, y) $\rightarrow$ THEME(e, x))
Example: Five Israelis, including two children, were killed yesterday.
THEME$(kill, Israeli)$ $\wedge$ PART-WHOLE$(child, Israeli)$ $\rightarrow$ THEME$(kill, child)$
Similar statements can be made for all the thematic roles: AGENT, EXPERIENCER, INSTRUMENT, CAUSE, LOCATION, etc.
AGENT ∘ ISA^{-1} = AGENT (AGENT(e, y) $\wedge$ ISA(x, y) $\rightarrow$ AGENT(e, x))
AGENT ∘ PART-WHOLE^{-1} = AGENT (AGENT(e, y) $\wedge$ PART-WHOLE(x, y) $\rightarrow$ AGENT(e, x))

Table 3: Examples of semantic combination axioms

ually checking all the LA Times corpus (w_1, w_3) pairs which satisfy $(R_i \circ R_j)(w_1, w_3)$. We have identified 64 semantic axioms that show how semantic relations can be combined. These axioms use relations such as PART-WHOLE, ISA, LOCATION, ATTRIBUTE, or AGENT. We listed several example rules in Table 3. The 64 axioms can be applied independent of the concepts involved in the semantic composition. We have also identified rules that can be applied only if the concepts that participate satisfy a certain condition or if the relations are of a certain type. For example, LOC ∘ LOC = LOC only if the LOC relation shows inclusion (*John is in the car in the garage* $\rightarrow$ LOC$(John, garage)$. *John is near the car behind the garage* $\not\rightarrow$ LOC$(John, garage)$).

4 FrameNet Can Help

The Berkeley FrameNet project[9] (Baker et al., 1998) is a lexicon-building effort based on the theory of *frame semantics* which defines the meanings of lexical units with respect to larger conceptual structures, called frames. Individual lexical units point to specific frames and establish a binding pattern to specific elements within the frame. FrameNet describes the underlying frames for different lexical units and examines sentences related to the frames using the BNC corpus. The result is an XML database that contains a set of frames, a set of frame elements for each frame, and a set of frame annotated sentences.

[9]http://framenet.icsi.berkeley.edu

4.1 Frame-based semantic axioms

With respect to a given target, the frame elements contribute to the understanding of the sentence. But they only link each argument to the target word (for example, THM$(theme, target)$ or AGT$(theme, target)$, LOC$(place, target)$, etc.). Often enough, we can find relations between the frame elements of a given frame. These new instances of semantic relations take as arguments the frame elements of a certain frame, when they are expressed in the text. For example, given the DEPARTING frame, we can say that the origin of the *theme* is the *source* (SRC$(theme, source)$) and that the new location of the *theme* is the *goal* frame element (LOC$(theme, goal)$). Moreover, if the text specifies the *cotheme* frame element, then we can make similar statements about it (SRC$(cotheme, source)$ and LOC$(cotheme, goal)$). These new relation instances increase the semantic information that can be derived from text.

So far, we manually inspected 54 frames and analyzed the relationships between their frame elements by examining their definitions and the annotated corpus provided with the FrameNet data. For each frame, we retained only the rules independent of the

CLOTHING_PARTS_F → PW($subpart, clothing$)
CLOTHING_PARTS_F → PW($material, subpart$) *Example:* “Hello, Hank” they said from the depths of the [fur]$_{Material}$ [collars]$_{Subpart,Target}$ of [their]$_{Wearer}$ [coats]$_{Clothing}$. PW($fur, collar$) and PW($collar, coat$)
CLOTHING_F → PAH($descriptor, garment$) ∨ PAH($descriptor, material$) *Example:* She didn’t bring heels with her so she decided on [gold]$_{Descriptor}$ [leather]$_{Material}$ [flip-flops]$_{Garment,Target}$. PAH($gold, leather$) ∨ PAH($gold, flip - flop$)
KINSHIP_F → KIN($ego, alter$) *Example:* The new subsidiary is headed by [Rupert Soames]$_{Alter}$, [son]$_{Target}$ [of the former British Ambassador to France and EC vice-president]$_{Ego}$. KIN(*Rupert Soames*, *the former British Ambassador to France and EC vice-president*)
GETTING_F → POS($recipient, theme$)
GETTING_F → ¬ POS($source, theme$) (only if the *source* is a person) *Example:* In some cases, [the BGS libraries]$_{Recipient}$ had [obtained]$_{Target}$ [copies of theses]$_{Theme}$ [from the authors]$_{Source}$ [by purchase or gift]$_{Means}$, and no loan records were available for such copies. POS(*the BGS libraries*, *copies of theses*) and ¬ POS($authors$, *copies of theses*)
GETTING_F → SRC($theme, source$) (if the *source* is not a person) *Example:* He also said that [Iran]$_{Recipient}$ [acquired]$_{Target}$ [fighter-bomber aircraft]$_{Theme}$ [from countries other than the USA and the Soviet Union]$_{Source}$. SRC(*fighter-bomber aircraft*, *countries other than the USA and the Soviet Union*)

Table 4: Frames-related semantic rules

frame’s lexical units. We identified 132 semantic axioms that hold in most cases[10]. We show some examples in Table 4.

4.2 Context importance

There are cases when the rules that we identified should not be applied. Let’s examine the sentence *John intends to leave the kitchen*. If we consider only the DEPARTING frame and its corresponding rules, without looking at the context, then our conclusions (¬ LOC($John, kitchen$) and SRC($John, kitchen$)) will be false. This sentence states an intention of motion, not the actual action. Therefore, our semantic axioms apply only when the context they are in, allows it. To overcome this problem, we do not apply the axioms for target words found in planning contexts, contexts related to beliefs, intentions, desires, etc. As an alternative, we keep track of plans, intentions, desires, etc. and, if, later on, we confirm them, then we apply the semantic axioms. Also, when we analyze a sentence, the frame whose rules we apply needs to be chosen carefully. For example, in the sentence [*A boat*]$_{Agent}$ [*carrying*]$_{Target}$ [*would-be Moroccan illegal emigrants*]$_{Theme}$ [*from UK*]$_{Path\_start}$ [*to Spain*]$_{Path\_end}$ *sank in the Strait of Gibraltar on June 8*, the CARRYING frame’s axioms do not apply. The boat nor the emigrants reach Spain (the *path_end* of the motion) because the boat sank. Here, the rules given by *sink.v*’s frame should be given priority over the *carry.v*’s rules. We can generalize and conclude that, given a sentence that contains more than one target (therefore, maybe multiple frames), the dominant frame, the one whose rules should be applied, is the frame given by the predicative verb. In the previous sentence, the dominant frame is the one given by *sink.v* and its rules should be applied before the axioms of the CARRYING frame. It should be noted that some of the axioms semantically related to the CARRYING frame still apply (for example, SRC($emigrants, UK$) or SRC($boat, UK$)). Unlike LOC($emigrants, Spain$), the previous relations do not conflict with the semantics given by *sink.v* and its location (*the Strait of Gibraltar*).

5 Experimental Results

5.1 The RTE data

The benchmark corpus for the RTE task consists of seven subsets with a 50%-50% split between the positive entailment examples and the negative ones. Each subgroup corresponds to a different NLP application: Information Retrival (IR), Comparable Documents (CD), Reading Comprehension (RC), Question Answering (QA), Information Extraction (IE), Machine Translation (MT), and Paraphrase Acquisition (PP). The RTE data set includes 1367 English (T, H) pairs from the news domain (political, eco-

[10]Section 4.2 lists some exception cases.

Semantic Axioms	**CD**		**IE**		**IR**		**MT**		**PP**		**QA**		**RC**	
	T	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**
Test data (%)														
applied to all Ts	13.33	21.33	26.66	10	6.66	4.44	11.66	10	8	0	15.38	7.69	21.43	17.14
applied to all Hs	1.33	9.33	5	10	0	0	1.66	1.66	8	0	1.53	0	0	1.43
solution for (T, H)	9.33	0	20	0	4.44	0	10	1.66	0	0	10.77	1.53	10	5.71
Development data (%)														
applied to all Ts	22	27.08	34.28	5.71	8.57	8.57	18.51	18.51	5.12	9.3	28.88	0	9.61	9.8
applied to all Hs	4	8.33	5.71	2.85	0	2.85	7.4	3.7	0	0	2.22	0	0	1.96
solution for (T, H)	10	2.08	22.85	5.71	5.71	2.85	18.51	3.7	2.56	0	20	0	7.69	0

Table 5: The impact of the semantic axioms on each NLP application data set. **T** and **F** stand for *True* and *False* entailments, respectively.

nomical, etc.). The development set consists of 567 examples and the test set contains the remaining 800 pairs.

5.2 Semantic axiom applicability

We measured the applicability of our set of semantic rules, by counting the number of times they extract new semantic information from text. Table 6 shows, in percentages, the coverage of the semantic axioms when applied to the texts T and the hypotheses H. We also show the number of times the semantic rules solve a (T, H) entailment without employing any other type of axioms.

Semantic Axioms	**True**	**False**	**Overall** (True and False)
Test data (%)			
applied to all Ts	15.75	11.75	13.75
applied to all Hs	2.00	3.74	2.87
both Ts and Hs	8.87	7.75	8.31
solution for (T, H)	10.25	1.50	5.87
Development data (%)			
applied to all Ts	18.02	11.26	14.64
applied to all Hs	2.47	2.81	2.65
both Ts and Hs	10.25	7.04	8.64
solution for (T, H)	12.36	1.76	7.05

Table 6: Applicability on the RTE data

Clearly, because the texts T convey much more information than H, they are the ones that benefit the most from our semantic axioms. The hypotheses H are more straightforward and a semantic parser can extract all their semantic information. Also, the rules tend to solve more positive (T, H) entailments. Because there are seven subsets corresponding to different NLP applications that make up the RTE data, we analyzed the contribution of our semantic axioms to each of the seven tasks. Table 5 shows the axioms' impact on each type of data. The logic-based approach proves to be useful to tasks like Information Extraction, Reading Comprehension, or Comparable Documents, and it doesn't seem to be the right choice for the more lexical-orientated applications like Paraphrase Acquisition, Machine Translation, and Information Retrieval.

5.3 RTE performance

To show the impact of our semantic axioms, we measured the contribution they bring to a system that participated in the RTE challenge. The ACC and F columns (Table 7) show the performance of the system before and after we added our semantic rules to the list of axioms needed by the logic prover.

Task	**Original**		**Enhanced**	
	ACC	F	ACC	F
Test-IR	.478	.472	.5	.505
Test-CD	.78	.736	.847	.819
Test-RC	.514	.558	.6	.636
Test-QA	.485	.481	.523	.537
Test-IE	.483	.603	.575	.687
Test-MT	.542	.444	.567	.49
Test-PP	.45	.585	.44	.576
Test	.551	.561	.604	.621
Development	.63	.619	.718	.714

Table 7: The accuracy(ACC) and f-measure(F) performance values of our system

The results show that richer semantic connectivity between text concepts improve the performance of a semantic entailment system. The overall accuracy increases with around 5% on the test data and almost 8% on the development set. We obtained performance improvements for all application settings, except for the Paraphrase Acquisition task. For this application, we obtained the smallest axiom coverage (Table 5). The impact of the semantic axioms on each NLP application data set correlates with the improvement that the addition of the rules brought to the system's accuracy.

Our error analysis showed that the system did not take full advantage of our semantic axioms, because the semantic parser did not identify all the semantic relations needed as building blocks by the axioms. We noticed a significant decrease in the logic prover's usage of world-knowledge axioms.

6 Conclusion

In this paper, we present a logic-based semantic approach for the *recognizing textual entailment* task. The system participating in the RTE competition used a set of world-knowledge, NLP, and lexical chain-based axioms and an in-house logic prover which received as input the logic forms of the two texts enhanced with semantic relation instances. Because the state-of-the-art semantic parsers cannot extract the complete semantic information encoded in text, the need for *semantic calculus* in NLP became evident. We introduce *semantic axioms* that either combine two semantic instances or label relations between the frame elements of a given frame. Preliminary statistical results show that incorporating semantic rules into the logic prover can double the semantic connectivity between the concepts of the analyzed text. Our process of identifying more semantic instances leads to a smaller dependency of the logic-based RTE system on world knowledge axioms, while improving its overall accuracy.

References

Collin Baker, Fillmore Charles, and John Love. 1998. The Berkeley FrameNet project. In *Proceedings of COLING/ACL*.

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 PASCAL RTE Challenge*.

Johan Bos and Katja Markert. 2005. Combining Shallow and Deep NLP Methods for Recognizing Textual Entailment. In *Proceedings of the PASCAL RTE Challenge*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL RTE Challenge*.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An Inference Model for Semantic Entailment in Natural Language. In *Proceedings of the PASCAL RTE Challenge*.

Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. 2005. Applying COGEX to Recognize Textual Entailment. In *Proceedings of the PASCAL RTE Challenge*.

Sanda Harabagiu and Dan Moldovan. 1998. Knowledge Processing on Extended WordNet. In *WordNet: an Electronic Lexical Database and Some of its Applications*.

Jess Herrera, Anselmo Peas, and Felisa Verdejo. 2005. Textual Entailment Recognision Based on Dependency Analysis and WordNet. In *Proceedings of the PASCAL RTE Challenge*.

Valentin Jijkoun and Maarten de Rijke. 2005. Recognizing Textual Entailment Using Lexical Similarity. In *Proceedings of the PASCAL RTE Challenge*.

Milen Kouylekov and Bernardo Magnini. 2005. Recognizing Textual Entailment with Tree Edit Distance Algorithms. In *Proceedings of the PASCAL RTE Challenge*.

Dan Moldovan and Adrian Novischi. 2002. Lexical Chains for Question Answering. In *Proceedings of COLING*.

Dan Moldovan and Vasile Rus. 2001. Logic Form Transformation of WordNet and its Applicability to Question Answering. In *Proceedings of ACL*.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. COGEX A Logic Prover for Question Answering. In *Proceedings of the HLT/NAACL*.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the Semantic Classification of Noun Phrases. In *Proceedings of HLT/NAACL, Computational Lexical Semantics workshop*.

Eamonn Newman, Nicola Stokes, John Dunnion, and Joe Carthy. 2005. UCD IIRG Approach to the Textual Entailment Challenge. In *Proceedings of the PASCAL RTE Challenge*.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher Manning, and Andrew Ng. 2005. Robust Textual Inference using Diverse Knowledge Sources. In *Proceedings of the PASCAL RTE Challenge*.

Detection of Entity Mentions Occurring in English and Chinese Text

Kadri Hacioglu, Benjamin Douglas and Ying Chen
Center for Spoken Language Research
University of Colorado at Boulder
{hacioglu,benjamin.douglas,yc}@colorado.edu

Abstract

In this paper, we describe an integrated approach to entity mention detection that yields a monolithic, almost language independent system. It is *optimal* in the sense that all categorical constraints are simultaneously considered. The system is compact and easy to develop and maintain, since only a single set of features and classifiers are needed to be designed and optimized. It is implemented using one-versus-all support vector machine (SVM) classifiers and a number of feature extractors at several linguistic levels. SVMs are well known for their ability to handle a large set of overlapping features with theoretically sound generalization properties. Data sparsity might be an important issue as a result of a large number of classes and relatively moderate training data size. However, we report results that the integrated system performs as good as a pipelined system that decomposes the problem into a few smaller subtasks. We conduct all our experiments using ACE 2004 data, evaluate the systems using ACE metrics and report competitive performance.

1 Introduction

The entity-relation (ER) model (Chen, 1976) views the physical world as a collection of entities with complex relationships. Automatic extraction of this model from raw text is important for creating a knowledge base (such as relational databases, marked-up text etc.) that can be used to achieve better end-to-end performances in several natural language processing (NLP) applications including information retrieval, question answering and machine translation. For example, in a typical QA system this knowledge base can be used to facilitate extraction of answers and retrieval of relevant documents.

Entities and relations in a document can be mentioned in several different ways. For example, a person entity, e.g. **Bill Clinton**, can be expressed in many different ways such as *The President*, *President Clinton*, *Mr. Clinton*, *he*, *him* etc. Similarly, one can express a geo-political entity, e.g. **United States**, as *his country* or another person entity, e.g. **Hillary Clinton**, as *his wife*, and their relation to the entity **Bill Clinton** as "president-of" and "family", respectively. It is clear that the detection of these mentions is the first crucial step for the extraction of the ER model to populate a database or an ontology.

Extraction of entities and their relationships is usually done in a pipelined system that first identifies entity mentions, next resolves mentions into unique entities (co-reference) and finally finds relations among them (Florian et al., 2004; Kambhatla, 2004). In that architecture, the errors in the first stage propagate and reduce the performance of subsequent stages; namely, co-reference resolver, that clusters all different mentions of an entity into a unique entity, and relation finder, that links entities according to their relationships. In fact, the subtask of entity mention detection itself is a very challeng-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 379–386, Vancouver, October 2005. ©2005 Association for Computational Linguistics

Table 1: Categorical structure of entities in ACE program

Entity Mention				
Entity			Mention	
Type	Sub-Type	Class	Type	Role

ing subtask since respective expressions can have relatively complex syntactic and categorical ("semantic") structures. That is, entity mentions in a body of text can occur in relatively complex embedded constructs with many attributes. Table 1 illustrates the categorical structure of an entity mention as specified in the Automatic Content Extraction (ACE) program run by NIST (ACE, 2004). Compared to the previous years the number of entity types and subtypes is greater.

The following segment of a sentence provides a typical example of the annotation:

[The [[Jordanian] military] spokesman] added ...

For simplicity, the entity mention attributes are excluded. The annotation clearly shows the embedded structure of entity mentions. We identify three entity mentions as *The Jordanian military spokesman*, *Jordanian military* and *Jordanian*.

Due to its complex nature, it is not uncommon that the mention detection task itself is also divided into a number of smaller sub-tasks. However, in this paper, we adopt an integrated classification approach to this problem that yields a monolithic structure. This allows all attributes, which define the categorical ("semantic") structure of a mention, to be jointly considered. The system has the ability to achieve better performance in principle provided that there is "enough" data to train, is easier to maintain and develop, and has a single set of features and classifiers to be engineered. All possible class labels are obtained by filling in the values of each attribute in the label **etype_subtype_class_mtype_role**, where, to avoid confusion, **etype** and **mtype** are used to denote entity and mention types, respectively.

Our data representation requires segmenting documents into sentences and then tokenizing sentences into words and punctuation. Each word is then assigned a label depending on its role in the mention. This data representation reduces the problem to a tagging task. For each token in focus, we create a number of features at lexical, syntactic and semantic levels. Additionally, we augment those features using features from external resources (e.g. named entity taggers, gazetteers, wordnet). We train a number of one-versus-all classifiers (Allwein et. al, 2000) using SVMs (Vapnik, 1995; Burges, 1998). During testing, classification of each token is performed in a greedy left-to-right manner using a finite-size sliding context window centered at the token in focus (Kudo and Matsumato, 2000).

This approach yields a large number of classes and a large number of overlapping features. We used a machine learning framework based on SVM classification since a large number of classes (in a one-versus-all set-up) and a large number of overlapping features can be easily handled with good generalization properties. We argue that data sparsity and computational complexity is not as severe as it might be expected in the other machine learning methods that are based on maximum likelihood parameter estimation. In other words, we claim that the large set of classification labels and training data sparseness are not major drawbacks. To provide evidence for this we also consider an approach that divides the task into relatively simpler tasks with considerably smaller numbers of labels. The approach yields a pipelined structure in which the decisions in earlier stages are used in later stages. We report results that the integrated approach performs similar to, and in some cases, even slightly better than the pipelined structure.

We also implement a novel post-processing scheme based on an entity base (EB) created from the tagged test data. This is motivated by the fact that an entity is identically referenced several times in a document. However, depending on the capitalization information of the entity mention and context in which it occurs, the entity can be missed at several positions in the document. A simple postprocessing algorithm that checks untagged tokens with low confidence against the EB is implemented. In doing so, it is highly likely that some of those missed entities could be identified. This is expected to reduce misses at the expense of false alarms. We report results that support our expectation.

The paper is organized as follows. Section 2 describes the ACE 2004 data used for training and evaluation. In Section 3, the problem is explained

Table 2: ACE 2004 corpus statistics for English and Chinese text.

Language	Train	Test
English	˜ 150K words	˜ 50K words
Chinese	˜ 150K words	˜ 50K words

and its data representation is introduced. Section 4 describes the general system architecture, that consists of a number of feature extractors, a (machine-learned) classifier and a simple post processor. In section 5, the features used for both English and Chinese systems are described. In section 6, we describe an alternative pipelined system. A novel post processing algorithm is introduced in section 7. Section 8 reports experimental results. Concluding remarks are made in the final section.

2 ACE Data

The ACE 2004 corpus consists of various text annotated for entities and relations. This corpus was created by the Linguistic Data Consortiom (LDC) in three languages: English, Chinese and Arabic (with support from the ACE program that began in 1999). Resources for data are newswire reports and broadcast news programs. Table 2 gives train and test statistics of this corpus for English and Chinese languages. Both languages have almost the same amount of data for both training and evaluation.

3 Problem Description and Data Representation

As shown in Table 1, an entity mention is characterized along 5 dimensions; namely *etype, sub-type class, mtype* and *role*. The ACE program specifies seven entity types; *person, organization, geo-political, location, facility, vehicle, weapon*. All entity types except *person* are further divided into several sub-types. For example, *organization* has *government, commercial, educational, non-profit* and *other* as its sub-types. The *class* attribute describes the kind of reference the entity mention makes to the entity in the world by taking one of the values {*generic, specific, negative, under-specified*} . Entity mentions are further characterized according to linguistic types of references as name (proper noun), nominal (common noun), pronominal (pronoun) and premodifier. The *role* of entity mention applies only to *geo-political* entities indicating the role of the entity in the context of the mention as one of *person, location, organization* and *geo-political*. For further details the reader is referred to (ACE, 2004)

All entity mentions in the original data are XML tagged with their respective attributes. In addition to the full extent of mentions, mention heads are also tagged. Referring to the previous example, the entity mention "The Jordanian military spokesman" which refers to a PERSON has the word "spokesman" as its head. Similarly, the entity mention "Jordanian military" which refers to an ORGANIZATION has the word "military" as its head. If one reduces the problem of entity mention detection to the detection of its head, the nature of the problem changes and the annotation of data becomes flat;

The [**GPE** Jordanian] [**ORG** military] [**PER** spokesman] .....

This allows us to consider the problem as a tagging/chunking problem and describe each word as beginning (B) an entity mention, inside (I) an entity mention or outside (O) an entity mention (Ramhsaw and Marcus, 1995; Sang and Veenstra, 1999). However, we believe that the information regarding the embedded structure in which the heads of entities occur is also useful for subsequent stages of an IE system including inference of relations among heads occurring in the same embedded construct. So, in addition to the IOB tags we introduce bracketing tags that might partially recover the embedded structure surrounding the heads. We refer to the following simple example

[Javier Trevino] was [the campaign manager for [the [ruling party] candidate [Fox] beat]].

to illustrate our tokenwise vertical representation:

#SNT_BEG#

Javier	B-PER_NAM
Trevino	I-PER_NAM
was	O

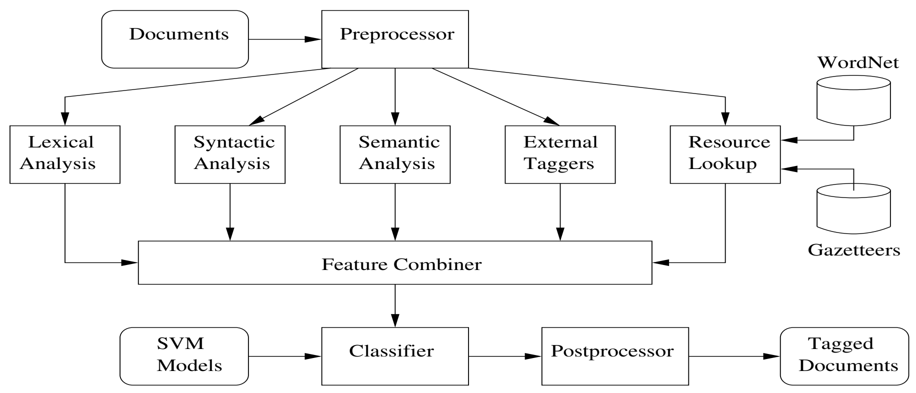


Figure 1: System Architecture

the	(*
campaign	*
manager	B-PER_NOM
for	*
the	(*
ruling	*
party	B-ORG_NOM
candidate	B-PER_NOM
Fox	B-PER_NAM
beat	*))
.	O
#SNT_END#	

If one does not use the bracketing representation, all non-head tokens will be labeled as ”Outside”. We believe that it is useful to discriminate the tokens that take part in mentions from those that do not occur in mentions.

4 General System Architecture

The general system block diagram is illustrated in Figure 1. It consists of a pre-processor, several feature extractors, a classifier and a post-processing module. Although the architecture is language independent, there are some minor language specific differences in some modules depending on the nature of the language and availability of resources for that language. In the following, we briefly describe both English and Chinese systems and indicate differences between them.

In the English system, the pre-processor segments the documents into sentences. It also includes a caser that restores the capitalization information of text without case (e.g. broadcast news) and a tokenizer that separates contractions and punctuation from words. Tokenized sentences are then processed at different linguistic levels to create features. At this stage, we employ a lexical pattern analyzer, part-of-speech tagger, a base phrase chunker, a syntactic parser, a dependency analyzer, look-up interfaces to external knowledge sources, and external small scale named entity taggers trained on different genres of text with different machine learning algorithms. All features are combined and then input to a classifier based on one-versus-all SVM classifiers. Finally, we perform simple post-processing to make sure that the final bracketing information is consistent.

The POS tagger and BP chunker are trained in-house using the Penn TreeBank. The syntactic parser is the Charniak parser which has models trained on the Penn TreeBank. The dependency analyzer performs dependency analysis using a set of head rules. The software was generously made available to us by the University of Maryland. The look-up interface to external knowledge sources such as WordNet or gazetteers is implemented using simple pattern matching.

In the Chinese system, the pre-processor is slightly different from that of the English system. It (obviously) does not need a caser and considers single Chinese characters as the minimal units of processing. It jointly segments a document into sentences and words. Then, it passes both word and sentence segmentation information to the subsequent stages along with Chinese characters. The SVM-based joint sentence/word segmenter is trained using the Chinese TreeBank (CTB). Linguistic analysis at different levels is performed in a manner similar to the analysis in the English system. In the Chinese system, the CTB is used to train a SVM-based POS tagger and BP chunker. The syntactic parser is trained on the CTB using Dan Bikel's parser. Dependency analysis is performed as in the English system using a set of Chinese head rules. Several in-house external taggers are trained using SVMs and different corpora. We have used only gazetteers for chinese as external knowledge sources.

5 Features

The following features are used in the English system:

- **tokens**: words in their original and all lower-cased forms
- **n-grams**: token prefixes and suffixes of length less than and equal to four
- **lexical patterns**: indicate case information (all lower-case, mixed case, first letter capital, all upper-case), is_hyphen, type (numeral, alphanumeral, alpha, other)
- **Part of Speech tags**
- **BP Positions**: The position of a token in a BP using the IOB representation (e.g. B-NP, I-NP, O etc.)
- **Clause tags**: The tags that mark token positions in a sentence with respect to clauses. (e.g *S)*S) marks a position that two clauses end)
- **Named entities-1**: The IOB tags of named entities. There are four categories; LOC, ORG, PERSON and MISC. A SVM-based tagger which is trained on CoNLL 2003 shared task data is used.
- **Named entities-2**: IOB tags of named entities found by the Identifinder (Bikel et. al, 1999); a HMM-based named entity tagger with 29 classes
- **Named entities-3**: IOB tags from a named entity tagger trained on MUC-6 and MUC-7 data using only the entity classes PERSON, LOCATION and ORGANIZATION.
- **Gazetteer labels**: indicate the name of the list to which the token belongs. Simple pattern matching is employed here.
- **WordNet categories**: concepts or class names in the WordNet 2.0 hypernym hierarchy rooted at "entity" concept. We trace hypernym hierarchies of the two most frequent senses of tokens that are tagged as nouns (NN, NNS, NNP etc.) to the top concepts. We count the number of concepts (that match to ACE entity types and subtypes) that occur in the hypernym hierarchy indicating that *token* is a (kind of) *concept*. The concepts (i.e entity/types/subtypes) with the maximum counts in the top two senses are selected as features (can also be considered as "maybe" labels)
- **Syntactic tags**: patterns of non-terminals and brackets that indicate the position of tokens in syntactic trees.
- **Head words**: words that the tokens depend
- **POS of Head words**:
- **main verb**: the verb at which the dependency parse tree is rooted.
- **Relations**: the grammatical and semantic relations between tokens and their heads.
- **Head word flag**: indicates whether the token plays a role of head in the sentence.

The features used in the Chinese system are

- **tokens**: Chinese characters
- **token positions**: IOB tags that indicate position of characters in words
- **Part of Speech tags**: POS tags of words to which tokens (characters) belong
- **BP Positions**: The position of a token in a BP using the IOB representation (e.g. B-NP, I-NP, O etc.)
- **Named entities-1**: IOB tags of two type of entities; *location* and *person*. A SVM based tagger trained on part of the Sinica corpus from Taiwan is used to generate these features.

- **Named entities-2**: IOB tags of named entities: person, location, organization etc. Another SVM based tagger trained on the People Daily data from mainland of China.

- **Gazetteer labels**: indicate the name of the list to which the token belongs. Simple pattern matching is employed here. Examples are labels that indicate Chinese last name, foreign person last name, first name etc.

- **Syntactic labels**: base phrase chunk labels and paths in syntactic trees

- **Head words**: as determined by Chinese dependency analysis

- **POS of Head words**:

- **Relations**: the grammatical and semantic relations between tokens and their heads.

6 A Pipelined System

As mentioned earlier the structure of entity mention categories is very complex. Considering all attributes together yields a large number of classes. One can argue that the large number of classes and data sparsity is an important issue here that it might have significant effect on performance. However, several attempts to divide the task into simpler subtasks have failed to yield a system with a better performance than that of the integrated system. In this section, we describe one such system.

The system consists of three stages in cascade: (i) entity mention extent detector, , (ii) mention type detector and (iii) entity type, subtype and mention role detector. Referring to the earlier example, the data representation in terms of class labels at each level is as follows:

#SNT_BEG#

Javier	(*	B-NAM	PER
Trevino	*)	I-NAM	PER
was	O	O	O
the	(*	O	O
campaign	*	O	O
manager	*	B-NOM	PER
for	*	O	O
the	(*	O	O
ruling	(*	O	O
party	*)	B-NOM	ORG
candidate	*)	B-NOM	PER
Fox	*	B-NAM	PER
beat	*))	O	O
.	O	O	O

#SNT_END#

where the second column is for the extent labels of mentions in bracketed representation, the third column is for the mention type labels in IOB representation and the last column is for the type labels (subtype and role labels are omitted for the sake of simplicity) of entity mentions in plain representation.

The pipelined system operates as follows. First it detects embedding structure of mention extents. Using that information the second stage identifies the type of mentions. In the final stage, the system identifies entity types, subtypes and mention roles using information (as features within context) from previous stages. Finally we combine all information into entity mention attributes and resolve inconsistencies by simple postprocessing.

Here, we have not done any feature selection specific to each stage. Instead we used the same features in all stages. One can argue that this is not the optimal set up for a cascaded system; separate feature design and selection should be made for each stage. Also we acknowledge that there are several other ways of dividing the task into smaller, simpler subtasks. Although we have not explored all possible pipelined architectures with all possible feature selections , we conjecture that the data sparsity is not as big an issue in SVMs as expected to be in the other machine learning algorithms based on maximum likelihood parameter estimation such as those based on maximum entropy (ME) or conditional random fields (CRF) frameworks.

7 A Novel Post-Processing Method

In our experiments, we have consistently observed that the identical mentions of a unique entity are missed depending on the missing capitalization information, unseen context and errors in feature extraction. For example, although the *name* mention of *person* “Eminem” is captured at several positions in the document, the entity mention “eminem” is missed, probably, due to its missing capitalization.

Table 3: Statistics on ACE 2004 data.

Language	Train Samples	Test Samples	# Joint Classes	# Pipelined Classes		
				Extent	MType	EType-SubTypey-Role
English	˜ 167K	˜ 61K	384	24	9	93
Chinese	˜ 307K	˜ 105K	374	15	7	95

As a solution we propose a post-processing method that is based on an entity base (EB) created from the tagged text. We populate the EB with all entity mentions (particularly with those that have *name* values) identified in the text. After we create the EB, we tag the text again by case insensitive pattern matching. We determine all tagged tokens that were initially left untagged or tagged with a different label by the SVM classifier. Using the SVM output (distance from separating hyperplane) as a confidence measure, we accept or reject the new tag based on a preselected threshold.

8 Experiments and Results

In this section, we describe the experiments conducted and results obtained using the ACE 2004 data. The number of training and test examples, which are words/punctuations in English and characters in Chinese, are summarized in Table 3. The number of classes in the joint task and in each pipelined subtask are also included.

In the first set of experiments we evaluated our integrated system and investigated the performance with respect to broad classes of features introduced in section 5, by adding one group of features at a time. Grouping of features into broad classes were done as follows:

- baseline features: tokens
- lexical features: POS, lexical patterns
- syntactic features: base phrase chunks, syntactic tree features
- "semantic" features: heads and grammatical relations
- external features: features from external resources; e.g. wordnet, gazetteers, other entity taggers etc.

Table 4: English system performance with respect to broad classes of features; lex: lexical features, syn: syntactic features, sem: "semantic" features, ext: external features, F_{uw}: unweighted F-score, F_w: weighted F-score, ACE: ACE value.

Feature class	F_{uw}	F_w	ACE
baseline (tokens)	56.5	54.8	36.1
baseline+lex	76.8	86.7	75.6
baseline+lex+syn	76.9	87.4	76.8
baseline+lex+syn+sem	77.1	87.8	77.6
baseline+lex+syn+sem+ext	82.0	90.7	82.9

The results are summarized in Table 4 and Table 5 for both English and Chinese systems. Both unweighted and weighted F-scores, and also ACE values are reported. It is interesting to note that significant gains were achieved by simple lexical and external features when they are added. The degree of improvement by using computationally intensive syntactic and dependency analysis is marginal. This might partly be due to the type of features derived from parse trees and partly due to the mismatch of the genre of text to the text on which the syntactic chunker and parser is trained. Since the dependency analysis is based on the syntactic analysis using a set of head rules, the extracted dependency based features might also be inaccurate. Although we observed moderate improvement for English, those features slightly hurt the performance of the Chinese system. This is because of the fact that the Chinese syntactic parser performs relatively worse than the English syntactic parser.

Table 6 presents the integrated and pipelined system performances using all features extracted for English and Chinese. Post-processing results are also included. It shows notable performance improvement with the recovery of many misses by post-processing. It should be noted that, in the pipelined architecture the post-processing is performed twice; at both mention and entity levels.

Table 5: Chinese system performance with respect to broad classes of features; lex: lexical features, syn: syntactic features, sem: "semantic" features, ext: external features, F_{uw}: unweighted F-score, F_w: weighted F-score, ACE: ACE value.

Feature class	F_{uw}	F_w	ACE
baseline (tokens)	77.6	83.5	70.8
baseline+lex	78.3	85.2	73.4
baseline+lex+syn	76.1	83.7	70.8
baseline+lex+syn+sem	74.8	83.6	70.8
baseline+lex+syn+sem+ext	78.4	86.8	76.1

9 Conclusions

We have discussed the significance of the entity mention detection in ER model extraction from raw text and presented the complex syntactic and categorical structure of the entity mentions specified in the ACE program. We have explored different ways of representing the problem and implemented two architecturally different (supervised) machine-learning based systems to accomplish the task; namely, a monolithic system and a cascaded system. We have described those systems in detail and empirically compared them. Both systems have achieved comparable performances on English text. However, the integrated system has achived moderately better performance on Chinese text. We have argued that it is easier to develop and maintain the monolithic system since it has a single set of features and classifiers to be tuned. We believe that the performance levels achieved at mid 80s (in ACE values) for English and at upper 70s for Chinese, using only the ACE data, are competitive. We have introduced a post-processing algorithm based on an entity base created during the testing. It has worked very well for both languages to recover several missed entity mentions and considerably improved the performance.

10 Acknowledgement

We extend our special thanks to Wayne Ward, Steven Bethard, James H. Martin and Dan Jurafsky for their useful feedback during this work. This work is supported by the ARDA Aquaint II Program via contract NBCHC040040.

Table 6: English and Chinese system performances with all features and post-processing: F_{uw}: unweighted F-score, F_w: weighted F-score, ACE: ACE value.

English System	F_{uw}	F_w	ACE
Integrated	82.0	90.7	82.9
Pipelined	82.1	90.8	83.1
Integrated+Post	82.2	91.5	**84.3**
Pipelined+Post	82.3	91.3	84.0

Chinese System			
Integrated	78.4	86.8	76.1
Pipelined	76.9	85.7	74.1
Integrated+Post	79.6	87.7	**77.5**
Pipelined+Post	79.1	86.6	75.6

References

E. L. Allwein, R. E Schapire and Y. Singer. 2000. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113-141,

Dan M. Bikel, Robert L. Schwartz, and Ralph M. Weischedel. 1999 An algorithm that learns what's in a name. *Machine Learning*, Vol. 34, pp. 211-231.

Chiristopher J. C. Burges 1998. Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2), pages 1-47.

Peter P. Chen 1976. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Trans. on Database Systems,* Vol. 1, No. 1, pages 1-36.

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. *Proceedings of HLT-2004*.

Nanda Kambhatla. 2004. Combining Lexical Syntactic and Semantic Features with Maximum Entropy Models for Extracting Relations. *Proceedings of ACL-04*.

Taku Kudo and Yuji Matsumato. 2000. Use of support vector learning for chunk identification. *Proc. of the 4th Conference on Very Large corpora*, pages 142-144.

Lance E. Ramhsaw and Mitchel P. Marcus. 1995. Text Chunking Using Transformation Based Learning. *Proceedings of the 3rd ACL Workshop on Very Large Corpora*, pages 82-94.

Erik F. T. J. Sangand and Jorn Veenstra 1999. Representing text chunks. *Proceedings of EACL'99*, pages 173-179.

The Automatic Content Extraction (ACE) Evaluation Plan. 2004. www.nist.gov/speech/tests/ace/

Vladamir Vapnik. 1995. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, USA.

Robust Textual Inference via Graph Matching

Aria D. Haghighi
Dept. of Computer Science
Stanford University
Stanford, CA
aria42@stanford.edu

Andrew Y. Ng
Dept. of Computer Science
Stanford University
Stanford, CA
ang@cs.stanford.edu

Christopher D. Manning
Dept. of Computer Science
Stanford University
Stanford, CA
manning@cs.stanford.edu

Abstract

We present a system for deciding whether a given sentence can be inferred from text. Each sentence is represented as a directed graph (extracted from a dependency parser) in which the nodes represent words or phrases, and the links represent syntactic and semantic relationships. We develop a learned graph matching approach to approximate entailment using the amount of the sentence's semantic content which is contained in the text. We present results on the Recognizing Textual Entailment dataset (Dagan et al., 2005), and show that our approach outperforms Bag-Of-Words and TF-IDF models. In addition, we explore common sources of errors in our approach and how to remedy them.

1 Introduction

A fundamental stumbling block for several NLP applications is the lack of robust and accurate semantic inference. For instance, question answering systems must be able to recognize, or infer, an answer which may be expressed differently from the query. Information extraction systems must also be able recognize the variability of equivalent linguistic expressions. Document summarization systems must generate succinct sentences which express the same content as the original document. In Machine Translation evaluation, we must be able to recognize legitimate translations which structurally differ from our reference translation.

One sub-task underlying these applications is the ability to *recognize* semantic entailment; whether one piece of text follows from another. In contrast to recent work which has successfully utilized logic-based abductive approaches to inference (Moldovan et al., 2003; Raina et al., 2005b), we adopt a graph-based representation of sentences, and use graph matching approach to measure the semantic overlap of text. Graph matching techniques have proven to be a useful approach for tractable approximate matching in other domains including computer vision. In the domain of language, graphs provide a natural way to express the dependencies between words and phrases in a sentence. Furthermore, graph matching also has the advantage of providing a framework for structural matching of phrases that would be difficult to resolve at the level of individual words.

2 Task Definition and Data

We describe our approach in the context of the 2005 Recognizing Textual Entailment (RTE) Challenge (Dagan et al., 2005), but note that our approach easily extends to other related inference tasks. The system presented here was one component of our research group's 2005 RTE submission (Raina et al., 2005a) which was the top-ranking system according to one of the two evaluation metrics.

In the 2005 RTE domain, we are given a set of pairs, each consisting of two parts: 1) the *text*, a

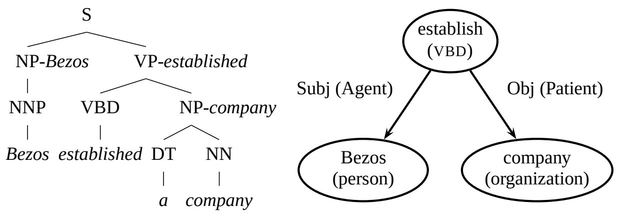


Figure 1: An example parse tree and the corresponding dependency graph. Each phrase of the parse tree is annotated with its head word, and the parenthetical edge labels in the dependency graph correspond to semantic roles.

small passage,[1] and the *hypothesis*, a single sentence. Our task is to decide if the hypothesis is "entailed" by the text. Here, "entails" does not mean strict *logical* implication, but roughly means that a competent speaker with basic world-knowledge would be happy to conclude the hypothesis given the text. This criterion has an aspect of relevance logic as opposed to material implication: while various additional background information may be needed for the hypothesis to follow, the text must substantially support the hypothesis.

Despite the informality of the criterion and the fact that the available world knowledge is left unspecified, human judges show extremely good agreement on this task – 3 human judges independent of the organizers calculated agreement rates with the released data set ranging from 91–96% (Dagan et al., 2005). We believe that this in part reflects that the task is fairly natural to human beings. For a flavor of the nature (and difficulty) of the task, see Table 1.

We give results on the data provided for the RTE task which consists of 567 development pairs and 800 test pairs. In both sets the pairs are divided into 7 tasks – each containing roughly the same number of entailed and not-entailed instances – which were used as both motivation and means for obtaining and constructing the data items. We will use the following toy example to illustrate our representation and matching technique:

Text: In 1994, Amazon.com was founded by Jeff Bezos.
Hypothesis: Bezos established a company.

[1]Usually a single sentence, but occasionally longer.

3 Semantic Representation

3.1 The Need for Dependencies

Perhaps the most common representation of text for assessing content is "Bag-Of-Words" or "Bag-of-N-Grams" (Papineni et al., 2002). However, such representations lose syntactic information which can be essential to determining entailment. Consider a Question Answer system searching for an answer to *When was Israel established?* A representation which did not utilize syntax would probably enthusiastically return an answer from (the 2005 RTE text): *The National Institute for Psychobiology in Israel was established in 1979.*

In this example, it's important to try to match relationships as well as words. In particular, any answer to the question should preserve the dependency between *Israel* and *established*. However, in the proposed answer, the expected dependency is missing although all the words are present.

Our approach is to view sentences as graphs between words and phrases, where dependency relationships, as in (Lin and Pantel, 2001), are characterized by the path between vertices.

Given this representation, we judge entailment by measuring not only how many of the *hypothesis* vertices are matched to the *text* but also how well the relationships between vertices in the hypothesis are preserved in their textual counterparts. For the remainder of the section we outline how we produce graphs from text, and in the next section we introduce our graph matching model.

3.2 From Text To Graphs

Starting with raw English text, we use a version of the parser described in (Klein and Manning, 2003), to obtain a parse tree. Then, we derive a dependency tree representation of the sentence using a slightly modified version of Collins' head propagation rules (Collins, 1999), which make main verbs not auxiliaries the head of sentences. Edges in the dependency graph are labeled by a set of hand-created `tgrep` expressions. These labels represent "surface" syntax relationships such as `subj` for subject and `amod` for adjective modifier, similar to the relations in *Minipar* (Lin and Pantel, 2001). The dependency graph is the basis for our graphical representation, but it is enhanced in the following ways:

Task	Text	Hypothesis	Entailed
Question Answer (QA)	Prince Charles was previously married to Princess Diana, who died in a car crash in Paris in August 1997.	Prince Charles and Princess Diana got married in August 1997.	False
Machine Translation (MT)	Sultan Al-Shawi, a.k.a the Attorney, said during a funeral held for the victims, ”They were all children of Iraq killed during the savage bombing.”.	The Attorney, said at the funeral, ”They were all Iraqis killed during the brutal shelling.”.	True
Comparable Documents (CD)	Napster, which started as an unauthorized song-swapping Web site, has transformed into a legal service offering music downloads for a monthly fee.	Napster illegally offers music downloads.	False
Paraphrase Recognition (PP)	Kerry hit Bush hard on his conduct on the war in Iraq.	Kerry shot Bush.	False
Information Retrieval (IR)	The country's largest private employer, Wal-Mart Stores Inc., is being sued by a number of its female employees who claim they were kept out of jobs in management because they are women.	Wal-Mart sued for sexual discrimination.	True

Table 1: Some Textual Entailment examples. The last three demonstrate some of the harder instances.

1. Collapse Collocations and Named-Entities: We "collapse" dependency nodes which represent named entities (e.g., *Jeff Bezos* in Figure fig-example) and also collocations listed in WordNet, including verbs and their adjacent particles (e.g., *blow_off* in *He blew off his work*) .

2. Dependency Folding: As in (Lin and Pantel, 2001), we found it useful to fold certain dependencies (such as modifying prepositions) so that modifiers became labels connecting the modifier's governor and dependent directly. For instance, in the text graph in Figure 2, we have changed *in* from a word into a relation between its head verb and the head of its NP complement.

3. Semantic Role Labeling: We also augment the graph representation with Probank-style semantic roles via the system described in (Toutanova et al., 2005). Each predicate adds an arc labeled with the appropriate semantic role to the head of the argument phrase. This helps to create links between words which share a deep semantic relation not evident in the surface syntax. Additionally, modifying phrases are labeled with their semantic types (e.g., *in 1991* is linked by a *Temporal* edge in the text graph of Figure 2), which should be useful in Question Answering tasks.

4. Coreference Links: Using a co-rereference resolution tagger, `coref` links are added throughout the graph. These links allowed connecting the referent entity to the vertices of the referring vertex. In the case of multiple sentence texts, it is our only "link" in the graph between entities in the two sentences.

For the remainder of the paper, we will refer to the text as T and hypothesis as H, and will speak of them in graph terminology. In addition we will use H_V and H_E to denote the vertices and edges, respectively, of H.

4 Entailment by Graph Matching

We take the view that a hypothesis is entailed from the text when the cost of matching the hypothesis graph to the text graph is low. For the remainder of this section, we outline a general model for assigning a match cost to graphs.

For hypothesis graph H, and text graph T, a *matching* M is a mapping from the vertices of H to those of T. For vertex v in H, we will use $M(v)$ to denote its "match" in T. As is common in statistical machine translation, we allow nodes in H to map to fictitious NULL vertices in T if necessary. Suppose the cost of matching M is Cost(M). If $\mathcal{M}$ is the set of such matchings, we define the cost of matching H to T to be

$$\text{MatchCost}(H,T) = \min_{M \in \mathcal{M}} \text{Cost}(M) \qquad (1)$$

Suppose we have a model, VertexSub$(v, M(v))$, which gives us a cost in $[0,1]$, for substituting vertex v in H for $M(v)$ in T. One natural cost model is to use the normalized cost for each of the vertex substitutions in M:

$$\text{VertexCost}(M) = \frac{1}{Z}\sum_{v \in H_V} w(v)\text{VertexSub}(v, M(v)) \tag{2}$$

Here, $w(v)$ represents the weight or relative importance for vertex v, and $Z = \sum_{v \in H_V} w(v)$ is a normalization constant. In our implementation, the weight of each vertex was based on the part-of-speech tag of the word or the type of named entity, if applicable. However, there are several other possibilities including using TF-IDF weights for words and phrases.

Notice that when $\text{Cost}(M)$ takes the form of (2), computing $\text{MatchCost}(H, T)$ is equivalent to finding the minimal cost bipartite graph-matching, which can be efficiently computed using linear programming.

We would like our cost-model to incorporate some measure of how relationships in H are preserved in T under M. Ideally, a matching should preserve all local relationships; i.e, if $v \rightarrow v' \in H_E$, then $M(v) \rightarrow M(v') \in T_E$. When this condition holds for all edges in H, H is isomorphic to a subgraph of T.

What we would like is an *approximate* notion of isomorphism, where we penalize the distortion of each edge relation in H. Consider an edge $e = (v, v') \in H_E$, and let $\phi_M(e)$ be the path from $M(v)$ to $M(v')$ in T.

Again, suppose we have a model, $\text{PathSub}(e, \phi_M(e))$ for assessing the "cost" of substituting a direct relation $e \in H_E$ for its counterpart, $\phi_M(e)$, under the matching. This leads to a formulation similar to (2), where we consider the normalized cost of substituting each edge relation in H with a path in T:

$$\text{RelationCost}(M) = \frac{1}{Z}\sum_{e \in H_E} w(e)\text{PathSub}(e, \phi_M(e)) \tag{3}$$

where $Z = \sum_{e \in H_E} w(e)$ is a normalization constant. As in the vertex case, we have weights for each hypothesis edge, $w(e)$, based upon the edge's label; typically subject and object relations are more important to match than others. Our final matching cost is given by a convex mixture of

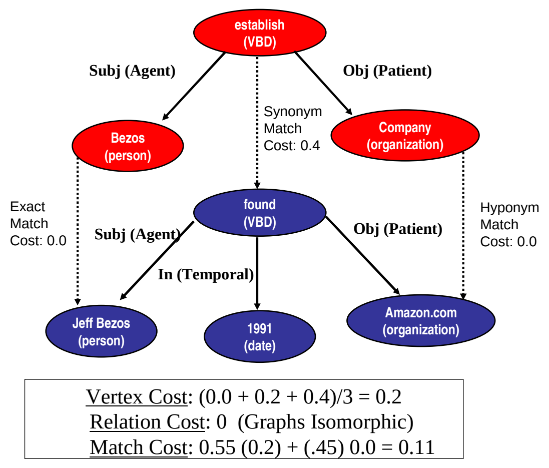


Figure 2: Example graph matching ($\alpha = 0.55$) for example pair. Dashed lines represent optimal matching.

the vertex and relational match costs: $\text{Cost}(M) = \alpha\text{VertexCost}(M) + (1 - \alpha)\text{RelationCost}(M)$.

Notice that minimizing $\text{Cost}(M)$ is computationally hard since if our PathSub model assigns zero cost only for preserving edges, then $\text{RelationCost}(M) = 0$ if and only if H is isomorphic to a subgraph of T. Since subgraph isomophism is an NP-complete problem, we cannot hope to have an efficient exact procedure for minimizing the graph matching cost. As an approximation, we can efficiently find the matching M^* which minimizes $\text{VertexCost}(\cdot)$; we then perform local greedy hill-climbing search, beginning from M^*, to approximate the minimal matching. The allowed operations are changing the assignment of any hypothesis vertex to a text one, and, to avoid ridges, swapping two hypothesis assignments

5 Node and Edge Substitution Models

In the previous section we described our graph matching model in terms of our VertexSub model, which gives a cost for substituting one graph vertex for another, and PathSub, which gives a cost for substituting the path relationship between two paths in one graph for that in another. We now outline these models.

5.1 Vertex substitution cost model

Our $\text{VertexSub}(v, M(v))$ model is based upon a sliding scale, where progressively higher costs are given based upon the following conditions:

- **Exact Match**: v and $M(v)$ are identical words/ phrases.
- **Stem Match**: v and $M(v)$'s stems match or one is a derivational form of the other; e.g., matching *coaches* to *coach*.
- **Synonym Match:** v and $M(v)$ are synonyms according to *WordNet* (Fellbaum, 1998). In particular we use the top 3 senses of both words to determine synsets.
- **Hypernym Match:** v is a "kind of" $M(v)$, as determined by *WordNet*. Note that this feature is asymmetric.
- **WordNet Similarity:** v and $M(v)$ are similar according to `WordNet::Similarity` (Pedersen et al., 2004). In particular, we use the measure described in (Resnik, 1995). We found it useful to only use similarities above a fixed threshold to ensure precision.
- **LSA Match:** v and $M(v)$ are distributionally similar according to a freely available Latent Semantic Indexing package,[2] or for verbs similar according to *VerbOcean* (Chklovski and Pantel, 2004).
- **POS Match:** v and $M(v)$ have the same part of speech.
- **No Match:** $M(v)$ is NULL.

Although the above conditions often produce reasonable matchings between text and hypothesis, we found the recall of these lexical resources to be far from adequate. More robust lexical resources would almost certainly boost performance.

5.2 Path substitution cost model

Our PathSub$(v \rightarrow v', M(v) \rightarrow M(v'))$ model is also based upon a sliding scale cost based upon the following conditions:

- **Exact Match:** $M(v) \rightarrow M(v')$ is an en edge in T with the same label.
- **Partial Match:** $M(v) \rightarrow M(v')$ is an en edge in T, not necessarily with the same label.
- **Ancestor Match:** $M(v)$ is an ancestor of $M(v')$. We use an exponentially increasing cost for longer distance relationships.

[2]Available at `http://infomap.stanford.edu`

- **Kinked Match:** $M(v)$ and $M(v')$ share a common parent or ancestor in T. We use an exponentially increasing cost based on the maximum of the node's distances to their least common ancestor in T.

These conditions capture many of the common ways in which relationships between entities are distorted in semantically related sentences. For instance, in our system, a partial match will occur whenever an edge type differs in detail, for instance use of the preposition *towards* in one case and *to* in the other. An ancestor match will occur whenever an indirect relation leads to the insertion of an intervening node in the dependency graph, such as matching *John is studying French farming* vs. *John is studying French farming practices*.

5.3 Learning Weights

Is it possible to learn weights for the relative importance of the conditions in the VertexSub and PathSub models? Consider the case where match costs are given only by equation (2) and vertices are weighted uniformly ($w(v) = 1$). Suppose that $\Phi(v, M(v))$ is a vector of features[3] indicating the cost according to each of the conditions listed for matching v to $M(v)$. Also let w be weights for each element of $\Phi(v, M(v))$. First we can model the substitution cost for a given matching as:

$$\text{VertexSub}(v, M(v)) = \frac{\exp\left(w^T \Phi(v, M(v))\right)}{1 + \exp\left(w^T \Phi(v, M(v))\right)}$$

Letting $s(\cdot)$ be the 1-sigmoid function used in the right hand side of the equation above, our final matching cost as a function of w is given by

$$c(H, T; w) = \min_{M \in \mathcal{M}} \frac{1}{|H_V|} \sum_{v \in H} s(w^T \Phi(v, M(v))) \tag{4}$$

Suppose we have a set of text/hypothesis pairs, $\{(T^{(1)}, H^{(1)}), \ldots, (T^{(n)}, H^{(n)})\}$, with labels $y^{(i)}$ which are 1 if $H^{(i)}$ is entailed by $T^{(i)}$ and 0 otherwise. Then we would like to choose w to minimize costs for entailed examples and maximize it for non-entailed pairs:

[3]In the case of our "match" conditions, these features will be binary.

$$\ell(w) = \sum_{i:y^{(i)}=1} \log c(H^{(i)}, T^{(i)}; w) + \sum_{i:y^{(i)}=0} \log(1 - c(H^{(i)}, T^{(i)}; w))$$

Unfortunately, $\ell(w)$ is not a convex function. Notice that the cost of each matching, M, implicitly depends on the current setting of the weights w. It can be shown that since each $c(H,T;w)$ involves minimizing $M \in \mathcal{M}$, which depends on w, it is not convex. Therefore, we can't hope to globally optimize our cost functions over w and must settle for an approximation.

One approach is to use coordinate ascent over M and w. Suppose that we begin with arbitrary weights and given these weights choose $M^{(i)}$ to minimize each $c(H^{(i)}, T^{(i)}; w)$. Then we use a relaxed form of the cost function where we use the matchings found in the last step:

$$\hat{c}(H^{(i)}, T^{(i)}; w) = \frac{1}{|H_V|} \sum_{v \in H} s(w^T \Phi(v, M^{(i)}(v)))$$

Then we maximize w with respect to $\ell(w)$ with each $c(\cdot)$ replaced with the cost-function $\hat{c}(\cdot)$. This step involves only logistic regression. We repeat this procedure until our weights converge.

To test the effectiveness of the above procedure we compared performance against baseline settings using a random split on the development set. Picking each weight uniformly at random resulted in 53% accuracy. Setting all weights identically to an arbitrary value gave 54%. The procedure above, where the weights are initialized to the same value, resulted in an accuracy of 57%. However, we believe there is still room for improvement since carefully-hand chosen weights results in comparable performance to the learned weights on the final test set. We believe this setting of learning under matchings is a rather general one and could be beneficial to other domains such as Machine Translation. In the future, we hope to find better approximation techniques for this problem.

6 Checks

One systematic source of error coming from our basic approach is the implicit assumption of upwards monotonicity of entailment; i.e., if T entails H then adding *more* words to T should also give us a sentence which entails H. This assumption, also made by other recent abductive approaches (Moldovan et al., 2003), does not hold for several classes of examples. Our formalism does not at present provide a general solution to this issue, but we include special case handling of the most common types of cases, which we outline below.[4] These checks are done after graph matching and assume we have stored the minimal cost matching.

Negation Check

Text: Clinton's book is not a bestseller

Hypothesis: Clinton's book is a bestseller

To catch such examples, we check that each hypothesis verb is not matched to a text word which is negated (unless the verb pairs are antonyms) and vice versa. In this instance, the *is* in H, denoted by *is*$_H$, is matched to *is*$_T$ which has a negation modifier, *not*$_T$, absent for *is*$_H$. So the negation check fails.

Factive Check

Text: Clonaid claims to have cloned 13 babies worldwide.

Hypothesis: Clonaid has cloned 13 babies.

Non-factive verbs (*claim*, *think*, *charged*, etc.) in contrast to factive verbs (*know*, *regret*, etc.) have sentential complements which do not represent true propositions. We detect such cases, by checking that each verb in H that is matched in T does not have a non-factive verb for a parent.

Superlative Check

Text: The Osaka World Trade Center is the tallest building in Western Japan.

Hypothesis: The Osaka World Trade Center is the tallest building in Japan.

In general, superlative modifiers (*most*, *biggest*, etc.) invert the typical monotonicity of entailment and must be handled as special cases. For any noun n with a superlative modifier (part-of-speech JJS) in H, we must ensure that all modifier relations of $M(n)$ are preserved in H. In this example, *building*$_H$ has a superlative modifier *tallest*$_H$, so we must ensure that each modifier relation of *Japan*$_T$, a noun

[4]All the examples are actual, or slightly altered, RTE examples.

Method	Accuracy	CWS
Random	50.0%	0.500
Bag-Of-Words	49.5%	0.548
TF-IDF	51.8%	0.560
GM-General	56.8%	0.614
GM-ByTask	56.7%	0.620

Table 2: Accuracy and confidence weighted score (CWS) for test set using various techniques.

dependent of *building*$_T$, has a *Western*$_T$ modifier not in H. So its fails the superlative check.

Additionally, during error analysis on the development set, we spotted the following cases where our VertexSub function erroneously labeled vertices as similar, and required special case consideration:

- **Antonym Check:** We consistently found that the `WordNet::Similarity` modules gave high-similarity to antonyms.[5] We explicitly check whether a matching involved antonyms and reject unless one of the vertices had a negation modifier.
- **Numeric Mismatch:** Since numeric expressions typically have the same part-of-speech tag (CD), they were typically matched when exact matches could not be found. However, mismatching numerical tokens usually indicated that H was not entailed, and so pairs with a numerical mismatch were rejected.

7 Experiments and Results

For our experiments we used the devolpement and test sets from the Recognizing Textual Entailment challenge (Dagan et al., 2005). We give results for our system as well as for the following systems:

- Bag-Of-Words: We tokenize the text and hypothesis and strip the function words, and stem the resulting words. The cost is given by the fraction of the hypothesis not matched in the text.
- TF-IDF: Similar to Bag-Of-Words except that there is a tf.idf weight associated with each hypothesis word so that more "important" words are higher weight for matching.

[5]This isn't necessarily incorrect, but is simply not suitable for textual inference.

Task	GM-General		GM-ByTask	
	Accuracy	CWS	Accuracy	CWS
CD	72.0%	0.742	76.0%	0.771
IE	55.9%	0.583	55.8%	0.595
IR	52.2%	0.564	51.1%	0.572
MT	50.0%	0.497	43.3%	0.489
PP	58.0%	0.741	58.0%	0.746
QA	53.8%	0.537	55.4%	0.556
RC	52.1%	0.539	52.9%	0.523

Table 3: Accuracy and confidence weighted score (CWS) split by task on the RTE test set.

We also present results for two graph matching (GM) systems. The GM-General system fits a single global threshold from the development set. The GM-ByTask system fits a different threshold for each of the tasks.

Our results are summarized in Table 2. As the result indicates, the task is particularly hard; all RTE participants scored between 50% and 60% in terms of overall accuracy (Dagan et al., 2005). Nevevertheless, both GM systems perform better than either Bag-Of-Words or TF-IDF. CWS refers to Confidence Weighted Score (also known as average precision). This measure is perhaps a more insightful measure, since it allows the inclusion of a ranking of answers by confidence and assesses whether you are correct on the pairs that you are most confident that you know the answer to. To assess CWS, our n answers are sorted in decreasing order by the confidence we return, and then for each i, we calculate a_i, our accuracy on our i most confident predictions. Then CWS $= \frac{1}{n}\sum_{i=1}^{n} a_i$.

We also present results on a per-task basis in Table 3. Interestingly, there is a large variation in performance depending on the task.

8 Conclusion

We have presented a learned graph matching approach to approximating textual entailment which outperforms models which only match at the word level, and is competitive with recent weighed abduction models (Moldovan et al., 2003). In addition, we explore problematic cases of nonmonotonicity in entailment, which are not naturally handled by either subgraph matching or the so-called "logic form"

Text	Hypothesis	True Ans.	Our Ans.	Conf	Comments
A Filipino hostage in Iraq was released.	A Filipino hostage was freed in Iraq.	True	True	0.84	Verb rewrite is handled. Phrasal ordering does not affect cost.
The government announced last week that it plans to raise oil prices.	Oil prices drop.	False	False	0.95	High cost given for substituting word for its antonym.
Shrek 2 rang up $92 million.	Shrek 2 earned $92 million.	True	False	0.59	Collocation "rang up" is not known to be similar to "earned".
Sonia Gandhi can be defeated in the next elections in India by BJP.	Sonia Gandhi is defeated by BJP.	False	True	0.77	"can be" does not indicate the complement event occurs.
Fighters loyal to Moqtada al-Sadr shot down a U.S. helicopter Thursday in the holy city of Najaf.	Fighters loyal to Moqtada al-Sadr shot down Najaf.	False	True	0.67	Should recognize non-Location cannot be substituted for Location.
C and D Technologies announced that it has closed the acquisition of Datel, Inc.	Datel Acquired C and D technologies.	False	True	0.64	Failed to penalize switch in semantic role structure enough

Table 4: Analysis of results on some RTE examples along with out guesses and confidence probabilities

inference of (Moldovan et al., 2003) and have proposed a way to capture common cases of this phenomenon. We believe that the methods employed in this work show much potential for improving the state-of-the-art in computational semantic inference.

9 Acknowledgments

Many thanks to Rajat Raina, Christopher Cox, Kristina Toutanova, Jenny Finkel, Marie-Catherine de Marneffe, and Bill MacCartney for providing us with linguistic modules and useful discussions. This work was supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) program.

References

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *EMNLP*.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop Recognizing Textual Entailment*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Knowledge Discovery and Data Mining*, pages 323–328.

Dan I. Moldovan, Christine Clark, Sanda M. Harabagiu, and Steven J. Maiorano. 2003. Cogex: A logic prover for question answering. In *HLT-NAACL*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ted Pedersen, Siddharth Parwardhan, and Jason Michelizzi. 2004. Wordnet::similarity – measuring the relatedness of concepts. In *AAAI*.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher D. Manning, and Andrew Y. Ng. 2005a. Robust textual inference using diverse knowledge sources. In *Proceedings of the First PASCAL Challenges Workshop*. Southampton, UK.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005b. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI 2005*. AAAI Press.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453.

Kristina Toutanova, Aria Haghighi, and Cristiopher Manning. 2005. Joint learning improves semantic role labeling. In *Association of Computational Linguistics (ACL)*.

Bootstrapping Without the Boot[*]

Jason Eisner and **Damianos Karakos**
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{eisner,damianos}@jhu.edu

Abstract

"Bootstrapping" methods for learning require a small amount of supervision to seed the learning process. We show that it is sometimes possible to eliminate this last bit of supervision, by trying many candidate seeds and selecting the one with the most plausible outcome. We discuss such "strapping" methods in general, and exhibit a particular method for strapping word-sense classifiers for ambiguous words. Our experiments on the Canadian Hansards show that our unsupervised technique is significantly more effective than picking seeds by hand (Yarowsky, 1995), which in turn is known to rival supervised methods.

1 Introduction

Some of NLP's most interesting problems have to do with unsupervised learning. Human language learners are able to discover word senses, grammatical genders, morphological systems, grammars, discourse registers, and so forth. One would like to build systems that discover the same linguistic patterns in raw text. For that matter, one would also like to discover patterns in bilingual text (for translation), in document collections (for categorization and retrieval), and in other data that fall outside the scope of humans' language learning.

There are relatively few successful methods for fully unsupervised learning from raw text. For example, the EM algorithm (Dempster et al., 1977) extracts the "wrong" patterns or gets stuck in local maxima.

One of the most promising avenues in recent years has been the use of "minimally supervised" methods. Such methods are initialized with some sort of "seed" that grows into a full classifier (or generative model). We say that a seed is "fertile" if it grows into a classifier (or model) that performs well on some desired criterion.

Ordinarily, it is up to a human to choose a seed that he or she intuitively expects to be fertile. While this may be easy when building a single classifier, it is prohibitive when building many classifiers. For example, we may wish to build

- word-sense classifiers for all words of a language (e.g., to get sharper lexical translation probabilities in a machine translation system)
- named-entity extractors for many languages
- new clusters or classifiers every day (for an evolving document collection)
- new clusters or classifiers every minute (for the document sets retrieved by *ad hoc* queries)
- many distinct classifiers that correspond to different views of the data[1]

Even when building a single classifier, a human may not know how to pick a good seed when working with an unfamiliar language or sublanguage, or when trying to induce less intuitive hidden variables, such as grammar rules or fine-grained senses. And there is no reason to expect humans to have good intuitions about seeds for mining non-linguistic data such as consumer purchasing records.

This paper considers how to remove this last element of supervision. Our idea is to guess a number of plausible seeds, build a classifier for each one, and then try to determine which of the seeds have grown successfully.

For example, to discover the two senses of the English word *drug*, we grow 200 classifiers (from different seeds) that attempt to partition instances of *drug* into two classes. We have no *direct* supervision about which of the resulting partitions corresponds to the true sense distinction. Instead, we rely on clues that tend to signal that a seed was fertile and led to a good partition. The clues are not specific to the word *drug*, but they may have been demonstrated to be good clues in general for successfully grown word sense disambiguators.

Demonstrated how? If we consider more than one clue, we may need some data to learn which clues to trust, and their relative weights. Our method is unsupervised in the conventional sense, as it obtains a classifier for *drug* with no supervision about *drug*. However, to learn what good classifiers generally look like[2] for this task, we first use

[*]We thank David Yarowsky for advice on the choice of data and for the *plant*/*tank* dataset.

[1]A word token or document can be characterized by a 20-bit vector, corresponding to its classifications by 20 different binary classifiers. These vectors are detailed abstract representations of the words or documents. They can be clustered, or all their bits can be included as potentially relevant features in another task.

[2]Ando and Zhang (2005) independently used this phrase, for a *semi-supervised, cross-task* learner that differs from our *unsupervised, cross-instance* learner. Both their work and ours try to transfer knowledge to a target problem from many artificial supervised "auxiliary problems," which are generated from unlabeled data (e.g., our pseudoword disambiguation problems). However, in their "structural learning," the target problem is *supervised* (if inadequately), and the auxiliary problems (supervised instances of a *different* task) are a source of useful *hidden features for the classifier*. In our "strapping," the target task is *unsupervised*, and the auxiliary problems (supervised instances

supervised data for a few *other* ambiguous words—or ambiguous pseudowords, a kind of artificial data where supervision comes for free. This supervision's effect on *drug* might be called *cross-instance learning*.

To take another metaphor, minimally supervised learning is often called "bootstrapping." Our goal is to allow a method to pull itself up by its own bootstraps[3] even when it has none. It places its stocking feet in anything handy, pulls on what it hopes to be sturdy straps, and checks to see how high it got.

We dub this family of methods "bootstrapping without the boot," or "strapping" for short. The name is meant to evoke "bagging" and "boosting"—other methods that train and combine multiple classifiers of the same form. However, we are careful to point out that strapping, unlike those theoretically motivated methods, is an *unsupervised* learning technique (in the sense explained above). The clusters or other hidden variables extracted by the winning classifier may or may not be the ones that one had hoped to find. Designing a strapping algorithm for a particular task requires more art than designing a supervised learner: one must invent not only appropriate features for classifying the data, but also appropriate clues for identifying "successful" classifiers.

2 Bootstrapping

To show where strapping might be useful, we briefly review a range of successful bootstrapping work. We consider different *tasks*. Given an *instance* of the task and a *seed* s for that instance, one bootstraps a classifier C_s that can classify *examples* of the task instance.

2.1 The Yarowsky algorithm

Yarowsky (1995) sparked considerable interest in bootstrapping with his successful method for word sense disambiguation. An instance of this task involves a homonymous word such as *drug*. A seed for the instance is a pair of words that are strongly associated, respectively, with the two senses of *drug*, such as (*trafficking*, *therapy*). An example is a token of *drug*.

For our purposes, a bootstrapping method can be regarded almost as a black box. However, we review the details of the Yarowsky algorithm to illustrate how bootstrapping is possible, and why some seeds are better than others. We will use these intuitions later in designing a method to strap the Yarowsky algorithm on a new instance—i.e., a method for *automatically* choosing seeds that discover a true sense distinction.

A learned classifier for the instance *drug* is an ordered decision list of contextual features (such as the presence of *dealer* nearby) that strongly indicate one or the other sense of *drug*. Given a sample token of *drug*, the classifier picks a sense according to the single highest-ranked feature that is present in the token's context.

To bootstrap a decision-list classifier from a seed, Yarowsky starts with all examples of *drug* that can be classified by using the seed words as the only features. These few examples are used as supervised data to train a longer decision list, which includes the seed words and any other features that suffice to distinguish these examples with high confidence. This longer decision list can now classify further examples, which are used to train a new and even longer decision list, and so on.

Yarowsky's method works if it can maintain high accuracy as it gradually increases its coverage. A precise classifier at iteration t tends to accurately classify new examples. This tends to produce a still-accurate classifier with greater coverage at iteration $t + 1$.

The method fails if the initial classifier is inaccurate (i.e., if the two seed words do not accurately pick out examples of the two senses). It may also fail if at some point, by bad luck on sparse data, the process learns some inappropriate features. If the classifier at iteration t is sufficiently polluted by bad features, the classifier at iteration $t + 1$ will start trying to distinguish examples that do *not* correspond to different senses, which may lead to even worse classifiers on subsequent iterations. However, some alternative seed may have escaped this bad luck by sprouting a different set of examples.

2.2 A Few Other Applications of Bootstrapping

Inspired by Yarowsky, Blum and Mitchell (1998) built a classifier for the task of web page classification.[4] They considered only one instance of this task, namely distinguishing course home pages from other web pages at a computer science department. Their seed consisted of 3 positive and 9 negative examples. Strapping a web page classifier would mean identifying seeds that lead to other "natural classes" of web pages. Strapping may be useful for unsupervised text categorization in general.

Riloff et al. (2003) learned lists of subjective nouns in English, seeding their method with 20 high-frequency, strongly subjective words. This seed set was chosen manually from an automatically generated list of 850 can-

of the *same* task) are a source of *clues for a meta-classifier* that chooses among classifiers grown from different seeds. In short, their auxiliary problems help train the target classifier directly, while ours help train only a simple meta-classifier that chooses among many unsupervised target classifiers. We use far fewer auxiliary problems but ours must be instances of the target task.

[3]The reference is to Baron Munchausen, a fictional 18th-century adventurer who rescued himself from a pit in this way. It is distinct from the "bootstrap" in non-parametric statistics.

[4]More precisely, they bootstrapped *two* Naive Bayes classifiers—one that looked at page content and the other that looked at links to the page. This "co-training" approach has become popular. It was also used by the Cucerzan and Yarowsky papers below, which looked at "internal" and "external" features of a phrase.

didate words. Strapping their method would identify subjective nouns in other languages, or other "natural classes" of English words.

Query expansion in IR searches for more documents "similar to" a designated relevant document. This problem too might be regarded as searching for a natural class—a small subset of documents that share some property of the original document—and approached using iterative bootstrapping. The seed would specify the original document *plus* one or two additional words or documents initially associated with the "relevant" and/or "irrelevant" classes. Strapping would guess various different seeds that extended the original document, then try to determine which seeds found a *cohesive* "relevant set."

Collins and Singer (1999) bootstrapped a system for classifying phrases in context. Again, they considered only one instance of this task: classifying English proper names as persons, organizations, or locations. Their seed consisted of 7 simple rules ("that *New York, California,* and *U.S.* are locations; that any name containing *Incorporated* is an organization; and that *I.B.M.* and *Microsoft* are organizations"). Strapping such a classifier would automatically discover named-entity classes in a different language, or other phrase classes in English.

Cucerzan and Yarowsky (1999) built a similar system that identified proper names as well as classifying them. Their seed consisted of a list of 40 to 300 names. Large seeds were not necessary for precision but did help recall.

Cucerzan and Yarowsky (2003) classified masculine vs. feminine nouns. They experimented with several task instances, namely different Indo-European languages. In each instance, their seed consisted of up to 30 feminine and 30 masculine words (e.g., *girl, princess, father*).

Many more papers along these lines could be listed. A rather different task is grammar induction, where a task instance is a corpus of text in some language, and the learned classifier is a parser. Following Chomsky (1981), we suggest that it may be possible to seed a grammar induction method with a small number of facts about the word order of the language: the basic clause order (SVO, SOV, etc.), whether pronominal subjects may be omitted (Chomsky's "*pro*-drop" parameter), etc. These facts can for example be used to construct a starting point for the inside-outside algorithm (Baker, 1979), which like other EM algorithms is highly sensitive to starting point. In a strapping method, one would guess a number of different seeds and evaluate the learned grammars on likelihood, entropy (Wang et al., 2002), correlation with semantics, or plausibility on other linguistic grounds that were not considered by the likelihood or the prior.

3 Strapping

Given a seed s for some task instance, let C_s denote the classifier grown from s. Let $f(s)$ denote the true fertility of a seed s, i.e., the performance of C_s measured against some set of correct answers for this instance. In general, we do not know the correct answers and hence do not know $f(s)$. That is why we are doing *unsupervised* learning.

Strapping relies on two *estimates* of $f(s)$. Let $g(s)$ be a quick estimate that considers only superficial features of the seed s. $h(s)$ is a more careful estimate that can be computed once C_s has been grown.

The basic method for strapping a classifier for a new task instance is very simple:

1. Quickly select a set S of candidate seeds such that $g(s)$ is high.
2. For each seed $s \in S$, learn a classifier C_s and measure $h(s)$.
3. Choose the seed $\hat{s} \in S$ that maximizes $h(\hat{s})$.
4. Return $C_{\hat{s}}$.

Variants on this method are obviously possible. For example, instead of returning a single classifier $C_{\hat{s}}$, one might use classifier combination to combine several classifiers C_s that have high $h(s)$.

It is clearly important that g and h be good estimates of f. Can data help us design g and h? Unfortunately, f is not known in an unsupervised setting. However, if one can get a few *supervised* instances of the same task, then one can select g and h so $g(s)$ and $h(s)$ approximate $f(s)$ for various seeds s for *those* instances, where $f(s)$ can be measured directly. The same g and h can then be used for *unsupervised* learning on all *new* task instances.

3.1 Selecting Candidate Seeds

The first step in strapping a classifier is to select a set S of seeds to try. For strapping to work, it is crucial that this set contain a fertile seed. How can this be arranged? Different strategies are appropriate for different problems and bootstrapping methods.

- Sometimes a simple heuristic $g(s)$ can help identify plausibly fertile seeds, as in the pseudocode above. In strapping the Yarowsky algorithm, we hope to find seeds $s = (x, y)$ such that x and y are strongly associated with different senses of the ambiguous target word. We choose $s = (x, y)$ such that x and y were never observed in the same sentence, but each of x and y has high pointwise mutual information with the ambiguous target word and appeared with it at least 5 times.
- If the space of possible seeds is small, it may be possible to try many or all of them. In grammar induction, for example, perhaps seeding with a few basic word order facts is enough. There are not so many basic word orders to try.

- Some methods have many fertile seeds—so many that a small random sample (perhaps filtered by $g(s)$) is likely to include at least one. We rely on this for the Yarowsky algorithm. If the target word is a true homonym, there exist many words x associated strongly with the first sense, and many words y associated strongly with the second sense. It is not difficult to stumble into a fertile seed $s = (x, y)$, just as it is not difficult for a human to think of one.[5]
- If fertile seeds are few and far between, one could abandon the use of a candidate set S selected by $g(s)$, and directly use general-purpose search methods to look for a seed whose predicted fertility $h(s)$ is high.

  For example, one could use genetic algorithms to breed a population of seeds with high $h(s)$. Or after evaluating several candidate seeds to obtain $h(s_1), h(s_2), \ldots h(s_k)$, one could perform a regression analysis that predicts $h(s)$ from superficial features of s, and use this regression function (a kind of $g(s)$ that is specific to the task instance) to pick s_{k+1}.

Strapping may be harder in cases like gender induction: it is hard to stumble into the kind of detailed seed used by Cucerzan and Yarowsky (2003). However, we suspect that fertile seeds exist that are much smaller than their lists of 50–60 words. While their large hand-crafted seed is sure to work, a handful of small seeds (each consisting of a *few* supposedly masculine and feminine words) might be likely to contain at least one that is fertile.[6] That would be sufficient, assuming we have a way to guess which seed in the handful is most fertile. That issue is at the core of strapping, and we now turn to it.

3.2 Clues for Evaluating Bootstrapped Classifiers

Once we have identified a candidate seed s and built the classifier C_s, we must evaluate whether C_s "looks like" the kind of classifier that tends to do well on our task.

This evaluation function $h(s)$ is task-specific. It may consider features of C_s, the growth trajectory of C_s, or the relation between C_s and other classifiers.

For concretness, we consider the Yarowsky method for word-sense disambiguation (WSD). How can we tell if a seed $s = (x, y)$ was fertile, without using even a small validation set to judge C_s? There are several types of *clues* to fertility, which may be combined into a meta-classifier that identifies fertile seeds.

Judge the result of classification with C_s: Even without a validation set, the result of running C_s on the training corpus can be validated in various ways, using independent plausibility criteria that were *not* considered by the bootstrapping learner.

- Is the classification reasonably balanced? (If virtually all examples of the target word are labeled with the same sense, then C_s has not found a sense distinction.)
- When a document contains multiple tokens of the target word, are all examples labeled with the same sense? This property tends to hold for correct classifiers (Gale et al., 1992a), at least for homonyms.
- True word senses usually correlate with document or passage topic. Thus, choose a measure of similarity between documents (e.g., the cosine measure in TF/IDF space). Does the target word tend to have the same sense in a document and in its nearby neighbors?
- True word senses may also improve performance on some task. Is the perplexity of a language model much reduced by knowing whether sense x or sense y (according to C_s) appeared in the current context? (This relates to the previous point.) Likewise, given a small bilingual text that has been automatically (and perhaps poorly) word-aligned, is it easier to predict how the target word will translate when we know its sense (according to C_s)?

Judge the internal structure of C_s: Does C_s look like a typical supervised decision list for word-sense disambiguation? For instance, does it contain many features with high log-likelihood ratios? (If a true sense distinction was discovered, we would expect *many* contextual features to correlate strongly with the predicted sense.)

Look at the process whereby C_s was learned: Does the bootstrapping run that starts from s look like a typical bootstrapping run from a fertile seed? For example, did it rapidly add many new examples with high confidence? Once new examples were classified, did their classifications remain stable rather than switching back and forth?

Judge the robustness of learning with seed s: Train several versions of C_s, as in ensemble methods (but unsupervised), by restricting each to a random subset of the data, or a subset of the available features. Do these versions tend to *agree* on how to classify the data? If not, seed s does not reliably find true (or even false) classes.

Judge the agreement of C_s with other classifiers: Are there several other classifiers $C_{s'}$ that agree strongly with C_s on examples that they both classify? If the sense

[5]Alignment methods in machine translation rely even more heavily on this property. While they begin with a small translation lexicon, they are sufficiently robust to the choice of this initial seed (lexicon) that it suffices to construct a single seed by crude automatic means (Brown et al., 1990; Melamed, 1997). Human supervision (or strapping) is unnecessary.

[6]This is particularly likely if one favors function words (in particular determiners and pronouns), which are strong indicators of gender. Cucerzan and Yarowsky used only content words because they could be extracted from bilingual dictionaries.

distinction is real, then many different seeds should be able to find it.

3.3 Training the Evaluation Function $h(s)$

Many of the above clues are necessary but not sufficient. For example, a learned classification may be robust without being a sense distinction. We therefore define $h(s)$ from a combination of several clues.

In general, $h(s)$ is a classifier or regression function that attempts to distinguish fertile from infertile seeds, given the clues. As mentioned earlier, we train its free parameters (e.g., coefficients for linear regression) on a few *supervised* instances of the task. These supervised instances allow us to measure the fertility $f(s)$ of various seeds, and thus to model the behavior of fertile versus infertile seeds. The presumption is that these behavior patterns will generalize to new seeds.

3.4 Training $h(s)$ on Artificial Data

Optionally, to avoid the need for any human annotation at all, the supervised task instances used to train $h(s)$ may be *artificial* instances, whose correct classifications are known without annotation.

In the case of word-sense disambiguation, one can automatically construct ambiguous *pseudowords* (Gale et al., 1992c; Schütze, 1998) by replacing all occurences of two words or phrases with their conflation. For example, *banana* and *wine* are replaced everywhere by *banana-wine*. The original, unconflated text serves as a supervised answer key for the artificial task of disambiguating *banana-wine*.

Traditionally, pseudowords are used as cheap test data to evaluate a disambiguation system. Our idea is to use them as cheap development data to tune a system. In our case, they tune a few free parameters of $h(s)$, which says what a good classifier for this task looks like. Pseudowords should be plausible instances of the task (Gaustad, 2001; Nakov and Hearst, 2003): so it is deliberate that *banana* and *wine* share syntactic and semantic features, as senses of real ambiguous words often do.

Cheap "pseudo-supervised" data are also available in some other strapping settings. For grammar induction, one could construct an artificial probabilistic grammar at random, and generate text from it. The task of recovering the grammar from the text then has a known answer.

4 Experiments

4.1 Unsupervised Training/Test Data

Our experiments focused on the original Yarowsky algorithm. We attempted to strap word-sense classifiers, using English data only, for English words whose French translations are ambiguous. This has obvious benefits for training an English-to-French MT system: separate parameters can be learned for the two senses of *drug*.[7]

Gale et al. (1992b) identified six such words in the Canadian Hansards, a parallel sentence-aligned corpus of parliamentary debate in English and French: *drug, duty, land, language, position, sentence*. We extracted all examples of each word from the 14-million-word English portion of the Hansards.[8] Note that this is considerably smaller than Yarowsky's (1995) corpus of 460 million words, so bootstrapping will not perform as well, and may be more sensitive to the choice of seed.

Because we are doing unsupervised learning, we both trained and tested these 6 words on the English Hansards. We used the French portion of the Hansards only to create a gold standard for evaluating our results.[9] If an English sentence containing *drug* is paired with a French sentence that contains exactly one of *médicament* or *drogue*, we take that as an infallible indicator of its sense.

4.2 Comparing Classifiers

Suppose binary classifier 1 assigns class "+" to a of n examples; binary classifier 2 assigns class "+" to b of the same n examples. Let e be the number of examples where the classifiers agree (both "+" or both "–").

An unsupervised classifier's polarity is arbitrary: classifier 1's "+" may correspond to classifier 2's "–". So we define the *overlap* as $E = \max(e, n - e)$, to reflect the best polarity.

To evaluate a learned classifier, we measure its overlap with the true classification. The statistical significance is the probability that this level of overlap would be reached by chance under independent classifications given the values a, b, n:

$$p = \sum_{\substack{\max(a+b-n,0) \,\le\, c \,\le\, \lfloor (a+b-E)/2 \rfloor \\ \text{or} \\ \lceil (a+b-(n-E))/2 \rceil \,\le\, c \,\le\, \min(a,b)}} \binom{a}{c} \binom{n-a}{b-c} \Big/ \binom{n}{b}$$

Also, we can measure the *agreement* between any two learned classifiers as $-(\log p)/n$. Note that a classifier that strongly favors one sense will have low agreement with other classifiers.

[7]To hedge against the possibility of misclassification, one could interpolate with non-sense-specific parameters.

[8]We are not certain that our version of the Hansards is identical to that in (Gale et al., 1992b).

[9]By contrast, Gale et al. (1992b) used the French portion as a source of training supervision. By contrast, we will assume that we do *not* have a large bilingual text such as the Hansards. We train only on the English portion of the Hansards, ignoring the French. This mimics the situation where we must construct an MT system with very little bilingual text. By first discovering word senses in unsupervised monolingual data (for either language), we can avoid incorrectly mixing up two senses of *drug* in our translation model.

4.3 Generating Candidate Seeds (via $g(s)$)

For each target word t, we chose candidate seeds $s = (x, y)$ with a high score $g(s)$, where $g(s) = \mathrm{MI}(t, x) + \mathrm{MI}(t, y)$, provided that $c(x, y) = 0$ and $c(t, x) \geq 5$ and $c(t, y) \geq 5$ and $1/9 < c(t, x)/c(t, y) < 9$.[10]

The set S of 200 seeds for t was constructed by repeatedly adding the top-scoring unused seed to S, except that to increase the variety of words, we disallowed a seed $s = (x, y)$ if x or y already appeared 60 times in S.

4.4 Hand-Picked Seeds

To compare, we chose two seeds by hand for each t.

The *casually* hand-picked seed was chosen by intuition from the list of 200 automatically generated seeds. This took about 2 minutes (per seed).

The *carefully* hand-picked seed was not limited to this list, and took up to 10 minutes to choose, in a data-guided fashion. We first looked at some supervised example sentences to understand the desired translational sense distinction, and then for each sense chose the highest-MI word that both met some stringent subjective criteria and appeared to retrieve an appropriate initial set of examples.

4.5 The Bootstrapping Classifier

Our approximate replication of Yarowsky's algorithm used only a small set of features:

- Original and lemmatized form of the word immediately preceding the target word t.
- Original and lemmatized form of the word immediately following t.
- Original and lemmatized form of the *content* words that appear in the same sentence as t.

We used the seed to provisionally classify any token of the target word that appeared in a sentence with exactly one of the two seed words. This formed our initial "training set" of disambiguated tokens. At each iteration of the algorithm, we trained a decision list on the current training set. We then used the decision list to reclassify all k tokens in the current training set, and also to augment the training set by classifying the *additional* $\max(50, k/10)$ tokens on which the decision list was most confident.[11]

[10] $c(x, y)$ counts the sentences containing both x and y. MI(t, x) $= \log c(t, x)c()/c(t)c(x)$ is pointwise mutual information.

[11] Such a token has some feature with high log-likelihood ratio, i.e., it strongly indicates one of the senses in the current training set. We smoothed using the method of (Yarowsky, 1996): when a feature has been observed with only one sense, its log-likelihood ratio is estimated as a linear function of the number of occurrences of the seen sense. Function words are smoothed with a different linear coefficient than content words, in order to discount their importance. We borrowed the actual coefficients from (Yarowsky, 1996), though we could have learned them.

4.6 Development Data (for tuning $h(s)$)

Before turning to the unsupervised Hansards, we tuned our fertility estimator $h(s)$ to identify good seeds on development data—i.e., on other, supervised task instances.

In the *supervised* condition, we used just 2 additional task instances, *plant* and *tank*, each with 4000 hand-annotated instances drawn from a large balanced corpus (Yarowsky, 1995).

In the *pseudo-supervised* condition, we used *no hand-annotated data*, instead constructing 10 artificial supervised task instances (section 3.4) from the English portion of the Hansards. To facilitate cross-instance learning, we tried to construct these pseudowords to behave something like our ambiguous test words.[12] Given a test word t, we randomly selected a seed (x, y) from its candidate list (section 4.3), excluding any that contained function words.[13] Our basic idea was to conflate x and y into a pseudoword x-y. However, to get a pseudoword with only two senses, we tried to focus on the particular senses of x and y that were selected by t. We constructed about 500 pseudoword tokens by using only x and y tokens that appeared in sentences that contained t, or in sentences resembling those under a TF-IDF measure. We repeated this process twice per test word to obtain 12 pseudowords. We then discarded the 2 pseudowords for which no seed beat baseline performance, reasoning that they were ill-chosen and unlike real ambiguous words.[14]

4.7 Clues to Fertility

For each seed s for each development or test target word, we measured a few clues $h_1(s), h_2(s) \ldots h_6(s)$ that we hoped might correlate with fertility. (In future work, we plan to investigate more clues inspired by section 3.2.)

- The *agreeability* of C_s with (some of) the other 199 classifiers:

$$\left(\frac{1}{199} \sum_{s' \neq s} \mathrm{agr}(C_s, C_{s'})^\gamma \right)^{1/\gamma}$$

  The agreement agr($C_s, C_{s'}$) was defined in section 4.2. We tried 4 values for γ (namely 1, 2, 5, 10), each resulting in a different feature.

[12] We used collocates of t. Perhaps better yet would be words that are distributionally similar to t (appear in same contexts). Such words tend to be syntactically and semantically like t.

[13] For an unknown language or domain, a lexicon of function words could be constructed automatically (Katz, 1996).

[14] Thus we discarded *alcohol-trafficking* and *addicts-alcohol*; note that these were indeed ill-chosen (difficult) since both words unluckily corresponded to the *same* sense of *drug*. This left us with *bound-constituents, customs-pray, claims-value, claims-veterans, culture-unparliamentary, english-learn, competitive-party, financial-party, death-quote, death-page.*

- The *robustness* of the seed, defined by the agreement of C_s with 10 variant classifiers $C_s^{(k)}$ that were trained with the same seed but under different conditions:

$$\frac{1}{10}\sum_{k=1}^{10} \text{agr}(C_s, C_s^{(k)})$$

  We simply trained each classifier $C_s^{(k)}$ on a random subset of the n test examples, chosen by sampling n times with replacement.[15]

- The *confidence* of C_s on its own training data: its average confidence over the n training tokens, minus the *classifier skew*.

  The decision list's confidence on a token is the log-likelihood ratio of the single feature used to classify that token. It has the form $|\log(c/d)|$ (perhaps smoothed) and was previously used to select data while bootstrapping C_s. Subtracting the skew, $|\log(a/(n-a))|$,[16] gives a measurement ≥ 0. It corrects for confidence that arises from the classifier's overall bias, leaving only the added value of the relevant contextual feature.

4.8 Tuning $h(s)$ and Strapping New Classifiers

For each of the 2 words or 10 pseudowords t in our development set (see section 4.6), we ranked its 200 seeds s by their true fertility $f(s)$. We then ran support vector regression[17] to learn a single linear function, $h(s) = \vec{w} \cdot$ (clue vector for C_s), that predicts the fertilities of all $2 \cdot 200$ or $10 \cdot 200$ seeds.[18]

Then, for each of our 6 Hansards test instances (section 4.1), we used $h(s)$ to pick the *top*-ranked of 200 seeds.[19] It took about 3 hours total to strap classifiers for all 6 instances, using about 40 machines and unoptimized Perl code on the 14-million-word Hansards. For each of the 6 instances, this involved selecting 200 candidate seeds, bootstrapping 11 classifiers $C_s, C_s^{(1)}, \ldots C_s^{(10)}$ from each seed, and choosing a particular C_s to return.

4.9 Results

Our results are in Table 1. On both development and test instances of the task, $g(s)$ proposed seeds with a good range of fertilities. The correlation of predicted with actual fertility on test data averaged an outstanding 85%.

Despite having no knowledge of the desired senses, strapping significantly beat human selection in *all 24* of the possible comparisons between a hand-picked seed (casual or careful) and a strapped seed (chosen by an $h(s)$ tuned on supervised or pseudo-supervised instances).

The $h(s)$ tuned on annotated *plant/tank* actually chose the *very best* of the 200 seeds in 4 of the 6 instances. The $h(s)$ tuned on artificial pseudowords did nearly as well, in 2 of 6 instances identifying the very best seed, and in 5 of 6 instances ranking it among its top 3 choices.

We conclude that our unsupervised clues to fertility actually work. Furthermore, combining clues via regression was wise, as it tended to work better than any single clue. Somewhat better regression weights for the WSD task were learned from 2 out-of-domain hand-annotated words than from 10 in-domain artificial pseudowords.

5 Open Questions

The work reported here raises many interesting questions for future research.

In the WSD task, we have only considered word types with two unrelated senses (homonyms). A more general problem is to determine when a word type is ambiguous at all, and if so, how many coarse-grained or fine-grained senses it has. Strapping seems naturally suited to this problem, since it aims to discover when a sense distinction grown from some seed is a *true* sense distinction.

Then we would like to know how well strapping generalizes to additional bootstrapping scenarios. Our WSD strapping experiments were successful using only a subset of the techniques proposed in section 3. Generalizing to other tasks may require other techniques for selecting and evaluating candidate seeds, and perhaps combining the resulting classifiers.

An interesting question is whether strapping can be used in an active learning context. Active learning is a kind of bootstrapping method that periodically requires new seeds: it turns to the user whenever it gets confused. Perhaps some of these seeds can be guessed nondeterministically and the guesses evaluated automatically, with or without user confirmation.

Finally, there may be theoretical guarantees about strapping when something is known about the data. When $h(s)$ is trained to estimate $f(s)$ well on some supervised instances, there may be guarantees about how strapping will perform on unsupervised instances drawn

[15]We eliminated duplicates, perhaps unfortunately.

[16]As before, a and $n - a$ are the numbers of tokens that C_s classifies as "+" and "–" respectively. Thus the skew is the log-likelihood ratio of the decision list's "baseline" feature.

[17]We used cross-validation among the 10 development pseudowords to choose the options to SVM*light* (Joachims, 1999): a linear kernel, a regularization parameter of 0.3, and a dependent variable of $10^{f(s)} \in [1, 10]$ rather than $f(s) \in [0, 1]$, which placed somewhat more emphasis on modeling the better seeds. Our development objective function was the average over the 10 pseudowords of the Spearman rank-order correlation between $h(s)$ and $f(s)$.

[18]We augmented the clue vector with binary clues of the form $t =$ *plant*, $t =$ *tank*, etc. The regression weight of such a clue is a learned bias term that models the inherent difficulty of the task instance t (which varies greatly by t). This allows the other regression features to focus on the quality of the seed *given* t.

[19]We do not have a clue $t = \ldots$ for this test instance. The resulting lack of a bias term may subtract a constant from the predicted fertilities—but that does not affect the ranking of seeds.

		drug		duty	land		language	position	sentence	
	baseline / # examples	51.2 / 371		70.1 / 633	76.6 / 1379		87.5 / 1012	81.7 / 2949	50.8 / 501	
	worst seed (of 200)	50.1 (200)	traffickers trafficking	50.0 (200)	50.1 (200)	claims farming	50.3 (200)	56.1 (200)	50.1 (200)	length life
	casually selected (from 200)	56.5 (87)	food trafficking	73.4^{*} (40)	76.2 (24)	farm veterans	86.4 (76)	81.7 (41)	80.6^{*} (40)	page prison
	carefully constructed	62.1^{*} (75)	alcohol costs	82.1^{*} (8.5)	76.6 (20)	farm strong	87.9 (25.5)	81.4 (56.5)	86.8^{*} (27)	death quote
	best/oracle seed (of 200)	$76.1^{*\dagger}$ (1)	alcohol medical	$86.2^{*\dagger}$ (1)	$81.3^{*\dagger}$ (1)	acres courts	$90.9^{*\dagger}$ (1)	$88.3^{*\dagger}$ (1)	$89.9^{*\dagger}$ (1)	reads served
strapping (unsupervised)	most agreeable seed ($\gamma = 1$)	$72.6^{*\dagger}$ (5)	abuse information	64.7 (47)	67.5 (36)	claims production	86.4 (79)	82.4 (36)	$88.7^{*\dagger}$ (10)	life quote
	most robust seed	**76.1***† (1)	alcohol medical	**86.2***† (1)	71.7 (29)	claims price	85.6 (93)	82.7 (21)	$88.8^{*\dagger}$ (9)	commuted next
	most confident seed	66.9^{*} (32)	trafficking used	72.1^{*} (42)	$77.9^{*\dagger}$ (3)	claims courts	$89.8^{*\dagger}$ (10)	$84.4^{*\dagger}$ (8)	**89.9***† (1)	reads served
	$h(s)$-picked (plant/tank)	**76.1***† (1)	alcohol medical	**86.2***† (1)	**81.3***† (1)	acres courts	$90.3^{*\dagger}$ (7)	$84.5^{*\dagger}$ (7)	**89.9***† (1)	reads served
	$h(s)$-picked (10 pseudowd)	$70.4^{*\dagger}$ (10)	alcohol found	**86.2***† (1)	$78.9^{*\dagger}$ (2)	children farm	$89.7^{*\dagger}$ (17)	$83.7^{*\dagger}$ (16)	**89.9***† (1)	reads served
	$h(s)$-picked, 2nd place	69.1^{*} (13)	alcohol related	$85.7^{*\dagger}$ (2)	$77.8^{*\dagger}$ (4)	aboriginal acres	**90.9***† (1)	82.8 (19)	$89.0^{*\dagger}$ (7)	prison quote
	$h(s)$-picked, 3rd place	**76.1***† (1)	alcohol medical	84.2^{*} (4)	$77.1^{*\dagger}$ (5)	acres cities	87.5 (28)	**88.3***† (1)	$88.6^{*\dagger}$ (15)	life reads
	$h(s)$ rank of oracle seed	3		1	14		2	3	1	
	Spearman rank-order corr.	0.863		0.905	0.718		0.825	0.842	0.937	

Table 1: [See section 4.9 for highlights.] Accuracy (as percentage) and rank (in parentheses) of bootstrapped classifiers for variously chosen seeds, some of which are shown. * denotes statistically significant agreement with the truth (section 4.2, $p < 0.01$). † denotes a seed having significantly better agreement with the truth than does the better of the hand-picked seeds (McNemar's test, $p < 0.03$). In each column, the best performance for an automatic or manual seed appears in **boldface**. The "most …" lines use no tuning, the "plant/tank" line tunes $h(s)$ on 2 supervised instances, and the subsequent lines tune $h(s)$ on 10 pseudoword instances. The last line gives the Spearman rank-order correlation between seeds' predicted fertilities $h(s)$ and their actual fertilities $f(s)$.

from the same source (cross-instance learning). Even in the fully unsupervised case, it may be possible to prove that if the data were generated from a particular kind of process (e.g., a Gaussian mixture), then a certain strapping algorithm can recover the hidden variables.

6 Conclusions

In this paper, we showed that it is sometimes possible—indeed, preferable—to eliminate the initial bit of supervision in "bootstrapping" algorithms such as the Yarowsky (1995) algorithm for word sense disambiguation. Our "strapping" approach tries many candidate seeds as starting points and evaluates them automatically. The evaluation function can be tuned if desired on other task instances, perhaps artificially constructed ones. It can then be used wherever human guidance is impractical.

We applied the method to unsupervised disambiguation of English words in the Canadian Hansards, as if for English-French translation. Our results (see section 4.9 for several highlights) show that our automatic "strapped" classifiers consistently outperform the classifiers bootstrapped from manually, knowledgeably chosen seeds.

References

R. K. Ando and T. Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication Papers Presented at the 97th meeting of the Acoustical Society of America*, MIT, Cambridge, MA, June.

A. Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*, July.

P. F. Brown, J. Cook, S.A. Della Pietra, V.G. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A statistical approach to machine translation. *CL*, 16(2).

N. Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of EMNLP/VLC*.

S. Cucerzan and D. Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proc. of EMNLP/VLC*.

S. Cucerzan and D. Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *Proc. of HLT/NAACL*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39(1):1–38.

W. A. Gale, K. W. Church, and D. Yarowsky. 1992a. One sense per discourse. In *Proc. of the 4th DARPA Speech and Natural Language Workshop*, pages 233–237.

W. A. Gale, K. W. Church, and D. Yarowsky. 1992b. Using bilingual materials to develop word sense disambiguation methods. In *Proc. of the 4th International Conf. on Theoretical and Methodological Issues in Machine Translation*.

W. A. Gale, K. W. Church, and D. Yarowsky. 1992c. Work on statistical methods for word sense disambiguation. In *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54–60.

T. Gaustad. 2001. Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *Proc. of ACL-EACL*.

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*. MIT Press.

S. M. Katz. 1996. Distribution of context words and phrases in text and language modelling. *NLE*, 2(1):15–59.

I. Dan Melamed. 1997. A word-to-word model of translational equivalence. In *Proc. of ACL/EACL*, page 490.

P. Nakov and M. Hearst. 2003. Category-based pseudowords. In *HLT-NAACL'03*, pages 67–69, Edmonton, Canada.

E. Riloff, J. Wiebe, and T. Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proc. of CoNLL*, pages 25–32, May–June.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 23.

S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. 2002. The latent maximum entropy principle. In *Proc. of ISIT*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

D. Yarowsky. 1996. *Three Machine Learning Algorithms for Lexical Ambiguity Resolution*. Ph.D. thesis, U. of Penn.

Differentiating Homonymy and Polysemy in Information Retrieval

Christopher Stokoe
School of Computing and Technology
University of Sunderland
UK
christopher.stokoe@sunderland.ac.uk

Abstract

Recent studies into Web retrieval have shown that word sense disambiguation can increase retrieval effectiveness. However, it remains unclear as to the minimum disambiguation accuracy required and the granularity with which one must define word sense in order to maximize these benefits. This study answers these questions using a simulation of the effects of ambiguity on information retrieval. It goes beyond previous studies by differentiating between homonymy and polysemy. Results show that retrieval is more sensitive to polysemy than homonymy and that, when resolving polysemy, accuracy as low as 55% can potentially lead to increased performance.

1 Introduction

Lexical ambiguity refers to words that share the same orthography but have different meanings (word senses). It can be sub-divided into two distinct types, homonymy and polysemy. Homonymy describes when two senses of a given word (or derivation) are distinct. Typically, they are separated by etymology and are therefore entirely unrelated in meaning. One classic example (Kilgarriff, 1992) is '*bat*' as in an airborne mammal (from the Middle English word '*bakke'* meaning flying rodent) vs. '*bat*' as in an instrument used in the game of cricket (from the Celtic for stick or cudgel). There is no underlying relationship between these two meanings which have come about independently from differing root languages. Alternatively, polysemy describes where two senses of a word are related in that they share membership of a subsuming semantic classification. Consider the word '*mouth*' as in a part of the body vs. '*mouth*' as in the outlet of a river. Both meanings are subsumed by a higher concept (in this case they both describe an opening). Homonymy and polysemy are differentiated in most dictionaries by the major (homonyms) and minor (polysemes) entries for a given word. Where a lexical resource is described in terms of granularity a coarse-grained approach only differentiates between homonymy whereas a fine-grained approach also considers polysemy.

The use of word sense disambiguation in Information Retrieval (IR) has been an active field of study for the past 30 years. Despite several failures (described in Sanderson, 2000) recent studies have begun to show increased retrieval effectiveness, particularly in Web retrieval. However, two key questions remain: (1) to what accuracy must disambiguation be performed in order to show increased retrieval effectiveness and (2) to what level of granularity should disambiguation be performed in order to maximize these gains? This study answers these questions by simulating the impact of ambiguity and its subsequent resolution on retrieval effectiveness.

2 Related Work

The motivation for this research is taken from recent studies (section 2.1) which have demonstrated increased retrieval effectiveness by accounting for word sense. The methodology is derived from previous studies (section 2.2) which model the impact that ambiguity and its subsequent resolution have on IR.

2.1 Accounting for Sense in IR

One of the first studies to show increased retrieval effectiveness through resolving ambiguity was Schütze and Pederson (1995). They used clustering to discriminate between alternate uses of a word. The clusters they produced were apparently fine-grained, although it is not clear if this observation was made with reference to a particular lexical resource. In terms of the accuracy to which they could discriminate meaning, a limited evaluation using a 10 word sample demonstrated accuracy approaching 90%. Results showed that retrieval effectiveness increased when documents were indexed by cluster as opposed to raw terms. Performance further increased when a word in the collection was assigned membership of its three most likely clusters. However, it is not clear if assigning multiple senses leads to coarser granularity or simply reduces the impact of erroneous disambiguation.

Stokoe et al. (2003) showed increased retrieval effectiveness through fine-grained disambiguation where a word occurrence in the collection was assigned one of the sense definitions contained in WordNet. The accuracy of their disambiguation was reported at 62% based on its performance over a large subset of SemCor (a collection of manually disambiguated documents). It remains unclear how accuracy figures produced on different collections can be compared. Stokoe et al. (2003) did not measure the actual performance of their disambiguation when it was applied to the WT10G (the IR collection used in their experiments). This highlights the difficulty involved in quantifying the effects of disambiguation within an IR collection given that the size of modern collections precludes manual disambiguation.

Finally, Kim et al. (2004) showed gains through coarse-grained disambiguation by assigning all nouns in the WT10G collection (section 3) membership to 25 top level semantic categories in WordNet (for more detail about the composition of WordNet see section 4). The motivation behind coarse-grained disambiguation in IR is that higher accuracy is achieved when only differentiating between homonyms. Several authors (Sanderson, 2000; Kim et al., 2004) postulate that fine-grained disambiguation may not offer any benefits over coarse-grained disambiguation which can be performed to a higher level of accuracy.

2.2 The Effects of Ambiguity on IR

The studies described in section 2.1 provide empirical evidence of the benefits of disambiguation. Unfortunately, they do not indicate the minimum accuracy or the optimal level of granularity required in order to bring about these benefits. Perhaps more telling are studies which have attempted to quantify the effects of ambiguity on IR.

Sanderson (1994) used pseudowords to add additional ambiguity to an IR collection. Pseudowords (Gale et al., 1992) are created by joining together randomly selected constituent words to create a unique term that has multiple controlled meanings. Sanderson (1994) offers the example of "*banana/kalashnikov*". This new term features two pseudosenses '*banana*' and '*kalashnikov*' and is used to replace any occurrences of the constituent words in the collection, thus introducing additional ambiguity. In his study, Sanderson experimented with adding ambiguity to the Reuters collection. Results showed that even introducing large amounts of additional ambiguity (size 10 pseudowords - indicating they had 10 constituents) had very little impact on retrieval effectiveness. Furthermore, attempts to resolve this ambiguity with less than 90% accuracy proved extremely detrimental.

Sanderson (1999) acknowledged that pseudowords are unlike real words as the random selection of their constituents ensures that the pseudosenses produced are unlikely to be related, in effect only modeling homonymy. Several studies (Schütze, 1998; Gaustad, 2001) suggest that this failure to model polysemy has a significant impact. Disambiguation algorithms evaluated using pseudowords show much better performance than when subsequently applied to real words. Gonzalo et al. (1998) cite this failure to model related senses in order to explain why their study into the effects of ambiguity showed radically different results to Sanderson (1994). They performed known item retrieval on 256 manually disambiguated documents and showed increased retrieval effectiveness where disambiguation was over 60% accurate. Whilst Sanderson's results no longer fit the empirical data, his pseudoword methodology does allow us to explore the effects of ambiguity without the overhead of manual disambiguation. Gaustad (2001) highlighted that the challenge lies in adapting pseudowords to account for polysemy.

Krovetz (1997) performed the only study to date which has explicitly attempted to differentiate between homonymy and polysemy in IR. Using the Longmans dictionary he grouped related senses based on any overlap that existed between two sense definitions for a given word. His results support the idea that grouping together related senses can increase retrieval effectiveness. However, the study does not contrast the relative merits of this technique against fine-grained approaches, thus highlighting that the question of granularity remains open. Which is the optimal approach? Grouping related senses or attempting to make fine-grained sense distinctions?

3 Experimental Setup

The experiments in this study use the WT10G corpus (Hawking and Craswell, 2002), an IR web test collection consisting of 1.69 million documents. There are two available Query / Relevance Judgments sets each consisting of 50 queries. This study uses the TREC 10 Web Track Ad-Hoc query set (NIST topics 501 – 550). The relevance judgments for these queries were produced using pooling based on the top 100 ranked documents retrieved by each of the systems that participated in the TREC 10 Web Track.

Initially the author produced an index of the WT10G and performed retrieval on this unmodified collection in order to measure baseline retrieval effectiveness. The ranking algorithm was length normalized TF.IDF (Salton and McGill, 1983) which is comparable to the studies in section 2. Next, two modified versions of the collection were produced where additional ambiguity in the form of pseudowords had been added. The first used pseudowords created by selecting constituent pseudosenses which are unrelated, thus introducing additional homonymy. The second used a new method of generating pseudowords that exhibit polysemy (the methodology is described in section 4.1). Contrasting retrieval performance over these three indexes quantifies the relative impact of both homonymy and polysemy on retrieval effectiveness. The final step was to measure the effects of attempting to resolve the additional ambiguity which had been added to the collection. In order to do this, the author simulated disambiguation to varying degrees of accuracy and measured the impact that this had on retrieval effectiveness.

4 Methodology

To date only Nakov and Hearst (2003) have looked into creating more plausible pseudowords. Working with medical abstracts (MEDLINE) and the controlled vocabulary contained in the MESH hierarchy they created pseudosense pairings that are related. By identifying pairs of MESH subject categories which frequently co-occurred and selecting constituents for their pseudowords from these pairings they produced a disambiguation test collection. Their results showed that category based pseudowords provided a more realistic test data set for disambiguation, in that evaluation using them more closely resembled real words. The challenge in this study lay in adapting these ideas for open domain text.

4.1 Pseudoword Generation

This study used WordNet (Miller et al., 1990) to inform the production of pseudowords. WordNet (2.0) is a hierarchical semantic network developed at Princeton University. Concepts in WordNet are represented by synsets and links between synsets represent hypernmy (subsumes) and hyponymy (subsumed) relationships in order to form a hierarchical structure. A unique word sense consists of a lemma and the particular synset in which that lemma occurs. WordNet is a fine-grained lexical resource and polysemy can be derived to varying degrees of granularity by traversing the link structure between synsets (figure 1).

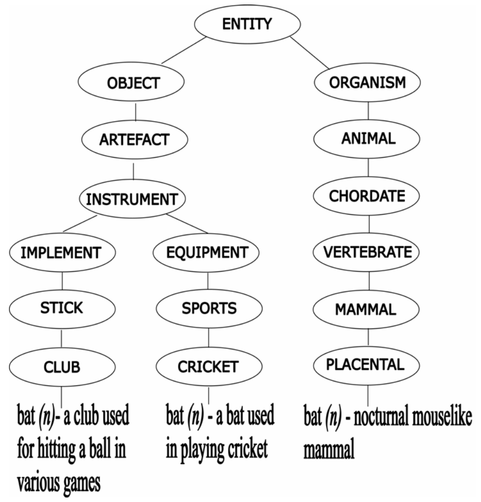

ENTITY
OBJECT
ORGANISM
ARTEFACT
ANIMAL
INSTRUMENT
CHORDATE
IMPLEMENT
EQUIPMENT
VERTEBRATE
STICK
SPORTS
MAMMAL
CLUB
CRICKET
PLACENTAL
bat (n)- a club used for hitting a ball in various games
bat (n) - a bat used in playing cricket
bat (n) - nocturnal mouselike mammal

Figure 1. A Subsection of the Noun Hierarchy in WordNet

An important feature of pseudowords is the number of constituents as this controls the amount of additional ambiguity created. A feature of all previous studies is that they generate pseudowords with a uniform number of constituents, e.g. size 2, size 5 or size 10, thus introducing uniform levels of additional ambiguity. It is clear that such an approach does not reflect real words given that they do not exhibit uniform levels of ambiguity. The approach taken in this study was to generate pseudowords where the number of constituents was variable. As each of the pseudowords in this study contain one query word from the IR collection then the number of constituents was linked directly to the number of senses of that word contained in WordNet. This effectively doubles the level of ambiguity expressed by the original query word. If a query word was not contained in WordNet then this was taken to be a proper name and exempted from the process of adding ambiguity. It was felt that to destroy any unambiguous proper names, which might act to anchor a query, would dramatically overstate the effects of ambiguity in terms of the IR simulation. The average size of the pseudowords produced in these experiments was 6.4 pseudosenses.

When producing the traditional pseudoword based collection the only modification to Sanderson's (1994) approach (described in section 2), other than the variable size, involved formalizing his observation that the constituent words were unlikely to be related. Given access to WordNet it was possible to guarantee that this is the case by rejecting constituents which could be linked through its inheritance hierarchy. This ensures that the pseudowords produced only display homonymy.

In order to produce pseudowords that model polysemy it was essential to devise a method for selecting constituents that have the property of relatedness. The approach taken was to deliberately select constituent words that could be linked to a sense of the original query word through WordNet. Thus the additional ambiguity added to the collection models any underlying relatedness expressed by the original senses of the query word. Pseudowords produced in this way will now be referred to as root pseudowords as this reflects that the ambiguity introduced is modeled around one root constituent. Consider the following worked example for the query "How are tornadoes formed?" After the removal of stopwords we are left with '*tornadoes*' and '*formed*' each of which is then transformed into a root pseudoword. The first step involves identifying any potential senses of the target word from WordNet. If we consider the word '*tornado*' it appears in two synsets:

> *1. tornado, twister -- (a localized and violently destructive windstorm occurring over land characterized by a funnel-shaped cloud extending toward the ground)*
>
> *2. crack, tornado -- (a purified and potent form of cocaine that is smoked rather than snorted)*

For each sense of the target word the system expands WordNet's inheritance hierarchy to produce a directed graph of its hypernyms. Figure 2 shows an example of this graph for the first sense of the word '*tornado*'. In order to ensure a related sense pair the system builds a pool of words which are subsumed by concepts contained in this graph. This is generated by recursively moving up the hierarchy until the pool contains at least one viable candidate. For a candidate to be viable it must meet the following criteria:

1) It must exist in the IR collection.
2) It must not be part of another pseudoword.
3) It can not be linked (through WordNet) to another constituent of the pseudoword.

The pool for sense 1 of '*tornado*' consists of [*hurricane*|*typhoon*], one of which is selected at random.

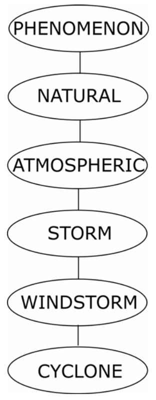


Figure 2. A Graph of the Hypernyms for the First Sense of the Word '*tornado*'

This process is repeated for each noun and verb sense of the query word. In this example there is one remaining sense of the word '*tornado*' - a slang term used to refer to the drug crack cocaine. For this sense the system produced a pool consisting of [*diacetylemorphine*|*heroin*]. Once all senses of the query word have been expanded the resulting pseudoword, '*tornadoes/hurricane/heroin*', is then used to replace all occurrences of its constituents within the collection. Through this process the system produces pseudowords with pseudosense pairings which have subsuming relationships, e.g. '*tornadoes/hurricane*' are subsumed by the higher category of '*cyclone*' whilst '*tornadoes/heroin*' are subsumed by the higher semantic category of '*hard_drug*'.

4.2 Pseudo-disambiguation

In order to perform pseudo-disambiguation the unmodified collection acts as a gold standard model answer. Through reducing each instance of a pseudoword back to one of its constituent components this models the selection process made by a disambiguation system. Obviously, the correct pseudosense for a given instance is the original word which appeared at that point in the collection. Variable levels of accuracy are introduced using a weighted probability model where the correct pseudosense for a given test instance is seeded with a fixed probability equivalent to the desired accuracy being simulated. When a disambiguation error is simulated one of the incorrect pseudosenses is selected randomly.

5 Results

The first set of results (section 5.1) addresses the question of granularity by quantifying the impact that adding either additional homonymy or polysemy has on retrieval effectiveness. The second set of results (section 5.2) looks at the question of disambiguation accuracy by simulating the impact that varied levels of accuracy have on retrieval effectiveness.

5.1 Homonymy vs. Polysemy

Let us first consider the impact of adding additional homonymy. Figure 3 graphs precision across the 11 standard points of recall for retrieval from both the baseline collection and one where additional homonymy has been added. Note that the introduction of additional homonymy brings about a small drop in retrieval effectiveness. With regard to the single value measures contained in table 1, this is a decrease of 2.5% in terms of absolute R-Precision (average precision after the total number of known relevant documents in the collection has been retrieved). This is a relative decrease of 14.3%. Similar drops in both precision@10 (precision after the first 10 documents retrieved) and average precision are also seen.

Next let us consider retrieval effectiveness over the root pseudoword collection where additional polysemy has been added (figure 4). Note that the introduction of additional polysemy has a more substantive impact upon retrieval effectiveness. In terms of R-Precision this decrease is 5.3% in absolute terms, a relative decrease of 30% compared to baseline retrieval from the unmodified collection. In addition, an even larger decrease in precision@10 occurs where the introduction of additional polysemy brings about a 7% drop in retrieval effectiveness.

In terms of the relative effects of homonymy and polysemy on retrieval effectiveness then note that adding additional polysemy has over double the impact of adding homonymy. This provides a clear indication that the retrieval process is more substantially affected by polysemy than homonymy.

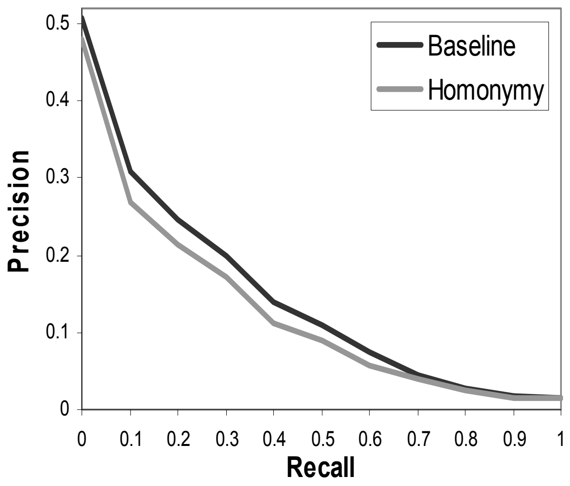


Figure 3. Precision across the 11 Standard Points of Recall for the Baseline and the Collection Containing Additional Homonymy

	R-Precision	Precision @10	Avg. Precision
Baseline	0.1732	0.2583	0.1334
Homonymy	0.1485	0.2208	0.1145
Polysemy	0.1206	0.1875	0.0951

Table 1. R-Precision, Precision@10 and Average Precision for all Three Versions of the Collection

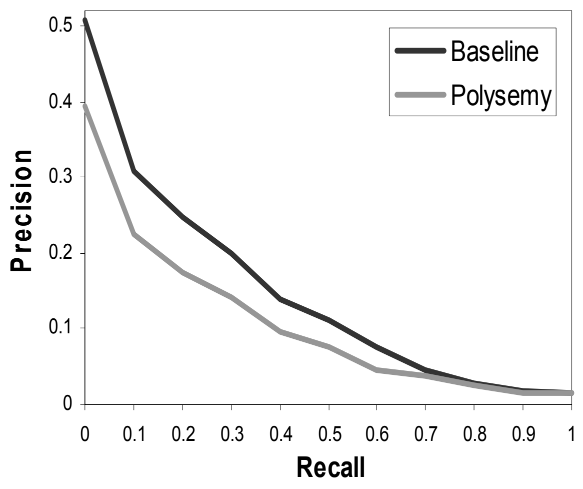


Figure 4. Precision across the 11 Standard Points of Recall for the Baseline and the Collection Containing Additional Polysemy

5.2 The Impact of Disambiguation

We now address the second part of the research question: to what accuracy should disambiguation be performed in order to enhance retrieval effectiveness? Figure 5 plots the impact, in terms of R-Precision, of performing disambiguation to varying degrees of accuracy after additional homonymy has been added to the collection. The dotted line represents the breakeven point, with R-Precision below this line indicating reduced performance as a result of disambiguation. Results show that where additional homonymy has been added to the collection disambiguation accuracy at or above 76% is required in order for disambiguation to be of benefit. Performing disambiguation which is less than 76% accurate leads to lower performance than if the additional homonymy had been left unresolved.

Moving on to consider the root pseudoword collection (figure 6) note that the breakeven point is only 55% where additional polysemy has been added. Consider that the results in section 5.1 showed that the introduction of additional polysemy had over double the impact of introducing additional homonymy. This is reflected in the relative effects of disambiguation in that the breakeven point is considerably lower for polysemy than homonymy.

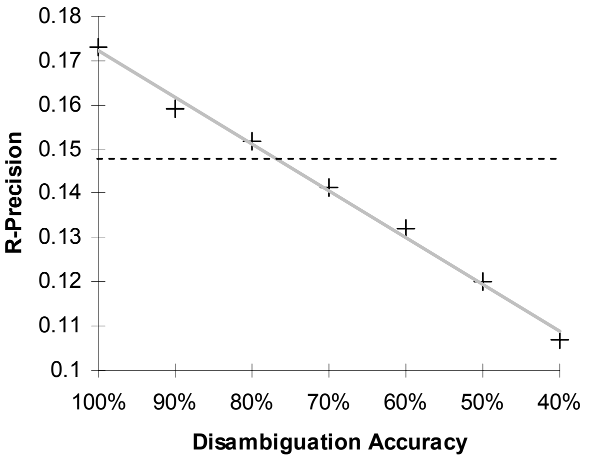


Figure 5. The Impact of Disambiguation on Effectiveness after the Addition of Homonymy

(Note the dashed line is the breakeven point)

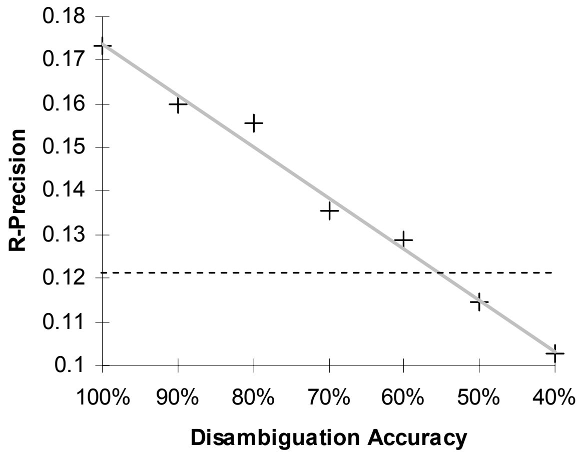


Figure 6. The Impact of Disambiguation on Effectiveness after the Addition of Polysemy

(Note the dashed line is the breakeven point)

6 Discussion

The results in section 5.1 show that retrieval effectiveness is more sensitive to polysemy than homonymy. One explanation for this can be hypothesized from previous studies (Krovetz and Croft, 1992; Sanderson and Van Rijsbergen, 1999) which highlight the importance of co-occurrence between query words. Where two (or more) words appear together in a query, statistical retrieval inherently performs some element of disambiguation. However, in the case of a word with many closely related senses, co-occurrence between query words may not be sufficient for a given sense to become apparent. This is particularly exasperated in Web retrieval given that the average query length in these experiments was 2.9 words. Clearly, the inherent disambiguation performed by statistical IR techniques is sensitive to polysemy in the same way as systems which explicitly perform disambiguation.

With regard to disambiguation accuracy and IR (section 5.2) these experiments establish that performance gains begin to occur where disambiguation is between 55% and 76%. Where within this range the actual breakeven point lies is dependent on the granularity of the disambiguation and the balance between polysemy and homonymy in a given collection. Consider that coarse-grained disambiguation is frequently advocated on the basis that it can be performed more accurately. Whilst this is undoubtedly true these results suggest that homonymy has to be resolved to a much higher level of accuracy than polysemy in order to be of benefit in IR.

It would seem prudent to consider the results of this study in relation to the state-of-the-art in disambiguation. At Senseval-3 (Mihalcea et al., 2004) the top systems were considered to have reached a ceiling, in terms of performance, at 72% for fine-grained disambiguation and 80% for coarse-grained. When producing the English language test collections the rate of agreement between humans performing manual disambiguation was approximately 74%. This suggests that machine disambiguation has reached levels comparable to the performance of humans. In parallel with this the IR community has begun to report increased retrieval effectiveness through explicitly performing disambiguation to varying levels of granularity.

A final point of discussion is the way in which we simulate disambiguation both in this study and those previously (Sanderson, 1994; Gonzalo et al., 1998). There is growing evidence (Leacock et al., 1998; Agirre and Martinez, 2004) to suggest that simulating uniform rates of accuracy and error across both words and senses may not reflect the performance of modern disambiguation systems. Supervised approaches are known to exhibit inherent bias that exists in their training data. Examples include Zipf's law (Zipf, 1949) which denotes that a small number of words make up a large percentage of word use and Krovetz and Croft's (1992) observation that one sense of a word accounts for the majority of all use. It would seem logical to presume that supervised systems show their best performance over the most frequent senses of the most frequent words in their training data. Not enough is known about the potential impact of these biases to allow for them to be incorporated into this simulation. Still, it should be noted that Stokoe et al. (2003) utilized frequency statistics in their disambiguator and that a by-product of Schütze and Pederson's (1992) approach was that they eliminated infrequently observed senses. There is supporting evidence from Sanderson and Van Rijsbergen (1999) to suggest that accounting for this frequency bias is in some way advantageous. Therefore, it is worth considering that simulating a uniform accuracy and error rate across all words and senses might actually offer a pessimistic picture of the potential for disambiguation and IR. Whilst this merits further study, the focus of this research was contrasting the relative effects of two types of ambiguity and both models were subject to the same uniform disambiguation.

7 Conclusions

This study has highlighted that retrieval systems are more sensitive to polysemy than homonymy. This leads the author to conclude that making fine-grained sense distinctions can offer increased retrieval effectiveness in addition to any benefits brought about by coarse-grained disambiguation. It also emphasises that although coarse-grained disambiguation can be performed to a higher degree of accuracy, this might not directly translate to increased IR performance compared to fine-grained approaches. This is in contrast to current thinking which suggests that coarse-grained approaches are more likely to bring about retrieval performance because of their increased accuracy.

In terms of disambiguation accuracy and increased retrieval effectiveness, results show potential benefits where accuracy is as low as 55% when dealing with just polysemy and rises to 76% when dealing with just homonymy. Obviously this study has simulated two extremes (polysemy or homonymy) and the exact point where performance increases will occur is likely to be dependent on the interaction between homonymy and polysemy in a given query.

With regard to simulation a more empirical exploration of the ideas expressed in this work would be desirable. However, the size of modern IR test collections dictates that future studies will need to rely more heavily on simulation. Therefore, until such time that a significant manually disambiguated IR collection exists pseudowords remain an interesting way to explore the effects of ambiguity within a large collection. The challenge lies in producing pseudowords that better model real words.

References

Agirre E. and Martinez D. 2004. *Unsupervised WSD based on automatically retrieved examples: the importance of bias*, in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). Barcelona, Spain.

Gale, W; Church, K. W; Yarowsky, D. 1992 *Work on Statistical Methods for Word Sense Disambiguation,* in Intelligent Probabilistic Approaches to Natural Language Papers, AAAI Press, Pp. 54 – 60.

Gaustad, T. 2001. *Statistical Corpus-based Word Sense Disambiguation: Pseudowords vs. Real Ambiguous Words,* in Companion Volume to the Proceedings of the 36th annual meeting of the ACL, Toulouse, France. Proceedings of the Student Research Workshop, ACL Press, Pp. 61 – 66.

Gonzalo, J; Verdejo, F; Chugur, I; Cigarran, J. 1998. *Indexing with WordNet Synsets Can Improve Text Retrieval,* in Proceedings of COLING / ACL '98 Workshop on the Usage of WordNet for NLP, Montreal, Canada, ACL Press, Pp. 38 – 44.

Hawking, D and Craswell, N. 2002. *Overview of the TREC-2001 Web Track,* in Proceedings of the 10th Text REtrieval Conference, Gaithersburg, MD. NIST Special Publication 500-250, Pp. 61 – 67.

Kilgarriff, A. 1992. *Polysemy*, Ph.D. Thesis, School of Cognitive and Computing Sciences, University of Sussex, Report CSRP 261

Kim, S; Seo, H; Rim, H. 2004. *Information Retrieval Using Word Senses: Root Sense Tagging Approach,* in Proceedings of the 27th ACM SIGIR, Sheffield, UK. Pp. 258 – 265.

Krovetz, R and Croft, W. B. 1992. *Lexical Ambiguity and Information Retrieval,* ACM Transactions on Information Systems, 10(2), Pp. 115 – 141.

Krovetz, R 1997. *Homonymy and Polysemy in Information Retrieval*, NEC Institute, Princeton.

Leacock, C; Chodorow, M; Miller, G. A. 1998 *Using Corpus Statistics and WordNet Relations for Sense Identification.* Computational Linguistics, 24(1), Pp. 147 – 165.

Mihalcea, R; Chklovski, T; Kilgarriff, A. 2004. *The Senseval-3 English Lexical Sample Task,* in Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, Pp. 25 – 28.

Miller, G; Beckwith, R; Fellbaum, C; Gross, D; Miller, K. 1990. *WordNet: An On-line Lexical Database,* International journal of lexicography, 3(4), Oxford University Press, Pp 235 – 244.

Nakov, P. and Hearst , M. 2003. *Category-based Pseudowords,* in Proceedings of the Human Language Technology Conference (HLT-NAACL 2003), Edmonton. Pp. 67–69.

Salton G and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*, McGraw-Hill.

Sanderson, M. 1994. *Word Sense Disambiguation and Information Retrieval,* in Proceedings of the 17th ACM SIGIR, Dublin, Ireland, ACM Press, Pp. 142 – 151.

Sanderson, M and Van Rijsbergen, C. J. 1999. *The Impact on Retrieval Effectiveness of Skewed Frequency Distributions,* ACM Transactions in Information Systems 17(4), ACM Press, Pp. 440 – 465.

Sanderson, M. 2000. *Retrieving with Good Sense* in Information Retrieval 2(1), Kluwer Academic Publishing, Pp. 49 – 69.

Schütze, H. 1998. *Automatic Word Sense Disambiguation,* Computational Linguistics 24(1), Pp. 113-120.

Schütze, H and Pederson, J. O. 1995. *Information Retrieval Based on Word Senses*, in Proceedings of the 4th Symposium on Document Analysis and Information Retrieval, Pp. 161 -175.

Stokoe, C. M; Oakes, M J; Tait, J I. 2003. *Word Sense Disambiguation in Information Retrieval Revisited* in Proceedings of the 26th ACM SIGIR, Toronto, Canada, Pp. 159 – 166.

Zipf, G. K. 1949. *Human Behavior and the Principle of Least-Effort*, Cambridge MA, U.S.A., Addison-Wesley

Unsupervised Large-Vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling

Rada Mihalcea
Department of Computer Science
University of North Texas
rada@cs.unt.edu

Abstract

This paper introduces a graph-based algorithm for sequence data labeling, using random walks on graphs encoding label dependencies. The algorithm is illustrated and tested in the context of an unsupervised word sense disambiguation problem, and shown to significantly outperform the accuracy achieved through individual label assignment, as measured on standard sense-annotated data sets.

1 Introduction

Many natural language processing tasks consist of labeling sequences of words with linguistic annotations, e.g. word sense disambiguation, part-of-speech tagging, named entity recognition, and others. Typical labeling algorithms attempt to formulate the annotation task as a traditional learning problem, where the correct label is *individually* determined for each word in the sequence using a learning process, usually conducted independent of the labels assigned to the other words in the sequence. Such algorithms do not have the ability to encode and thereby exploit dependencies across labels corresponding to the words in the sequence, which potentially limits their performance in applications where such dependencies can influence the selection of the correct set of labels.

In this paper, we introduce a graph-based sequence data labeling algorithm well suited for such natural language annotation tasks. The algorithm simultaneously annotates all the words in a sequence by exploiting relations identified among word labels, using random walks on graphs encoding label dependencies. The random walks are mathematically modeled through iterative graph-based algorithms, which are applied on the label graph associated with the given sequence of words, resulting in a stationary distribution over label probabilities. These probabilities are then used to simultaneously select the most probable set of labels for the words in the input sequence.

The annotation method is illustrated and tested on an unsupervised word sense disambiguation problem, targeting the annotation of all open-class words in unrestricted text using information derived exclusively from dictionary definitions. The graph-based sequence data labeling algorithm significantly outperforms the accuracy achieved through individual data labeling, resulting in an error reduction of 10.7%, as measured on standard sense-annotated data sets. The method is also shown to exceed the performance of other previously proposed unsupervised word sense disambiguation algorithms.

2 Iterative Graphical Algorithms for Sequence Data Labeling

In this section, we introduce the iterative graphical algorithm for sequence data labeling. The algorithm is succinctly illustrated using a sample sequence for a generic annotation problem, with a more extensive illustration and evaluation provided in Section 3.

Given a sequence of words $W = \{w_1, w_2, ..., w_n\}$, each word w_i with corresponding admissible labels $L_{w_i} = \{l^1_{w_i}, l^2_{w_i}, ..., l^{N_{w_i}}_{w_i}\}$, we define a label graph G = (V,E) such that there is a vertex $v \in V$ for every possible label $l^j_{w_i}$, $i = 1..n$, $j = 1..N_{w_i}$. Dependencies between pairs of labels are represented as directed or indirected edges $e \in E$, defined over the set of vertex pairs $V \times V$. Such label dependencies can be learned from annotated data, or derived by other means, as illustrated later. Figure 1 shows an example of a graph-

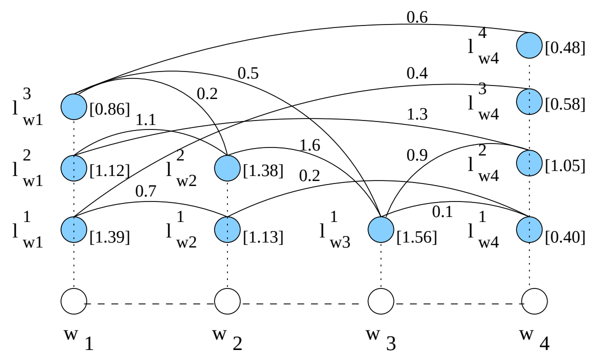


Figure 1: Sample graph built on the set of possible labels (shaded nodes) for a sequence of four words (unshaded nodes). Label dependencies are indicated as edge weights. Scores computed by the graph-based algorithm are shown in brackets, next to each label.

ical structure derived over the set of labels for a sequence of four words. Note that the graph does not have to be fully connected, as not all label pairs can be related by a dependency.

Given such a label graph associated with a sequence of words, the likelihood of each label can be recursively determined using an iterative graph-based ranking algorithm, which runs over the graph of labels and identifies the importance of each label (vertex) in the graph. The iterative graphical algorithm is modeling a random walk, leading to a stationary distribution over label probabilities, represented as scores attached to vertices in the graph. These scores are then used to identify the most probable label for each word, resulting in the annotation of all the words in the input sequence. For instance, for the graph drawn in Figure 1, the word w_1 will be assigned with label $l^1_{w_1}$, since the score associated with this label (1.39) is the maximum among the scores assigned to all admissible labels associated with this word.

A remarkable property that makes these iterative graphical algorithms appealing for sequence data labeling is the fact that they take into account global information recursively drawn from the entire graph, rather than relying on local vertex-specific information. Through the random walk performed on the label graph, these iterative algorithms attempt to collectively exploit the dependencies drawn between *all* labels in the graph, which makes them superior to other approaches that rely only on local information, individually derived for each word in the sequence.

2.1 Graph-based Ranking

The basic idea implemented by an iterative graph-based ranking algorithm is that of “voting” or “recommendation”. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting a vote determines how important the vote itself is, and this information is also taken into account by the ranking algorithm.

While there are several graph-based ranking algorithms previously proposed in the literature, we focus on only one such algorithm, namely PageRank (Brin and Page, 1998), as it was previously found successful in a number of applications, including Web link analysis, social networks, citation analysis, and more recently in several text processing applications.

Given a graph $G = (V, E)$, let $In(V_a)$ be the set of vertices that point to vertex V_a (predecessors), and let $Out(V_a)$ be the set of vertices that vertex V_a points to (successors). The PageRank score associated with the vertex V_a is then defined using a recursive function that integrates the scores of its predecessors:

$$P(V_a) = (1 - d) + d * \sum_{V_b \in In(V_a)} \frac{P(V_b)}{|Out(V_b)|} \qquad (1)$$

where d is a parameter that is set between 0 and 1[1].

This vertex scoring scheme is based on a random walk model, where a walker takes random steps on the graph G, with the walk being modeled as a Markov process – that is, the decision on what edge to follow is solely based on the vertex where the walker is currently located. Under certain conditions, this model converges to a stationary distribution of probabilities, associated with vertices in the graph. Based on the Ergodic theorem for Markov chains (Grimmett and Stirzaker, 1989), the algorithm is guaranteed to converge if the graph is both aperiodic and irreducible. The first condition is achieved for any graph that is a non-bipartite graph, while the second condition holds for any strongly connected graph – property achieved by PageRank through the random jumps introduced by the $(1 - d)$ factor. In matrix notation, the PageRank vector of stationary probabilities is the principal eigenvector for the matrix A_{row}, which is obtained from the adjacency matrix A representing the graph, with all rows normalized to sum to 1: ($P = A^T_{row} P$).

Intuitively, the stationary probability associated with a vertex in the graph represents the probability

[1]The typical value for d is 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation.

of finding the walker at that vertex during the random walk, and thus it represents the importance of the vertex within the graph. In the context of sequence data labeling, the random walk is performed on the label graph associated with a sequence of words, and thus the resulting stationary distribution of probabilities can be used to decide on the most probable set of labels for the given sequence.

2.2 Ranking on Weighted Graphs

In a weighted graph, the decision on what edge to follow during a random walk is also taking into account the weights of outgoing edges, with a higher likelihood of following an edge that has a larger weight. The weighted version of the ranking algorithm is particularly useful for sequence data labeling, since the dependencies between pairs of labels are more naturally modeled through weights indicating their strength, rather than binary $0/1$ values. Given a set of weights w_{ab} associated with edges connecting vertices V_a and V_b, the weighted PageRank score is determined as:

$$WP(V_a) = (1-d)+d \sum_{V_b \in In(V_a)} \frac{w_{ba}}{\sum_{V_c \in Out(V_b)} w_{bc}} WP(V_b) \quad (2)$$

2.3 Algorithm for Sequence Data Labeling

Given a sequence of words with their corresponding admissible labels, the algorithm for sequence data labeling seeks to identify a graph of label dependencies on which a random walk can be performed, resulting in a set of scores that can be used for label assignment. Algorithm 1 shows the pseudocode for the labeling process. The algorithm consists of three main steps: (1) construction of label dependencies graph; (2) label scoring using graph-based ranking algorithms; (3) label assignment.

First, a weighted graph of label dependencies is built by adding a vertex for each admissible label, and an edge for each pair of labels for which a dependency is identified. A maximum allowable distance can be set ($MaxDist$), indicating a constraint over the distance between words for which a label dependency is sought. For instance, if $MaxDist$ is set to 3, no edges will be drawn between labels corresponding to words that are more than three words apart, counting all running words. Label dependencies are determined through the $Dependency$ function, whose definition depends on the application and type of resources available (see Section 2.4).

Next, scores are assigned to vertices using a graph-based ranking algorithm. Current experiments are based on PageRank, but other ranking algorithms can be used as well.

Algorithm 1 Graph-based Sequence Data Labeling

Input: Sequence $W = \{w_i | i = 1..N\}$
Input: Admissible labels $L_{w_i} = \{l^t_{w_i} | t = 1..N_{w_i}\}$,$i = 1..N$
Output: Sequence of labels $L = \{l_{w_i} | i = 1..N\}$, with label l_{w_i} corresponding to word w_i from the input sequence.

Build graph G of label dependencies

```
 1: for $i = 1$ to $N$ do
 2:   for $j = i + 1$ to $N$ do
 3:     if $j - i > MaxDist$ then
 4:       $break$
 5:     end if
 6:     for $t = 1$ to $N_{w_i}$ do
 7:       for $s = 1$ to $N_{w_j}$ do
 8:         $weight \leftarrow Dependency(l^t_{w_i}, l^s_{w_j}, w_i, w_j)$
 9:         if $weight > 0$ then
10:           $AddEdge(G, l^t_{w_i}, l^s_{w_j}, weight)$
11:         end if
12:       end for
13:     end for
14:   end for
15: end for
```

Score vertices in G

```
 1: repeat
 2:   for all $V_a \in Vertices(G)$ do
 3:     $WP(V_a) = (1 - d) + d*$
          $\sum_{V_b \in In(V_a)} w_{ba} WP(V_b) / \sum_{V_c \in Out(V_b)} w_{bc}$
 4:   end for
 5: until convergence of scores $WP(V_a)$
```

Label assignment

```
 1: for $i = 1$ to $N$ do
 2:   $l_{w_i} \leftarrow argmax\{WP(l^t_{w_i}) | t = 1..N_{w_i}\}$
 3: end for
```

Finally, the most likely set of labels is determined by identifying for each word the label that has the highest score. Note that all admissible labels corresponding to the words in the input sequence are assigned with a score, and thus the selection of two or more most likely labels for a word is also possible.

2.4 Label Dependencies

Label dependencies can be defined in various ways, depending on the application at hand and on the knowledge sources that are available. If an annotated corpus is available, dependencies can be defined as label co-occurrence probabilities approximated with frequency counts $P(l^t_{w_i}, l^s_{w_j})$, or as conditional probabilities $P(l^t_{w_i} | l^s_{w_j})$. Optionally, these dependencies can be lexicalized by taking into account the corresponding words in the sequence, e.g. $P(l^t_{w_i} | l^s_{w_j}) \times P(w_i | l^t_{w_i})$. In the absence of an annotated corpus, dependencies can be derived by other means, e.g. part-of-speech probabilities can be approximated from a raw corpus as in (Cutting et al., 1992), word-sense dependencies can be derived as definition-based similarities, etc. Label dependencies are set as weights on the arcs drawn between corresponding labels. Arcs can be directed or undirected for joint probabilities or similarity measures, and are usually directed for conditional probabilities.

2.5 Labeling Example

Consider again the example from Figure 1, consisting of a sequence of four words, and their possible corresponding labels. In the first step of the algorithm, label dependencies are determined, and let us assume that the values for these dependencies are as indicated through the edge weights in Figure 1. Next, vertices in the graph are scored using an iterative ranking algorithm, resulting in a score attached to each label, shown in brackets next to each vertex. Finally, the most probable label for each word is selected. Word w_1 is thus assigned with label $l^1_{w_1}$, since the score of this label (1.39) is the maximum among the scores associated with all its possible labels (1.39, 1.12, 0.86). Similarly, word w_2 is assigned with label $l^2_{w_2}$, w_3 with label $l^1_{w_3}$, and w_4 receives label $l^2_{w_4}$.

2.6 Efficiency Considerations

For a sequence of words $W = \{w_1, w_2, ..., w_n\}$, each word w_i with N_{w_i} admissible labels, the running time of the graph-based sequence data labeling algorithm is proportional with O($C \sum_{i=1}^{n} \sum_{j=i+1}^{i+MaxDist} (N_{w_i} \times N_{w_j})$) (the time spent in building the label graph and iterating the algorithm for a constant number of times C). This is order of magnitudes better than the running time of O($\prod_{i=1}^{n} N_{w_i}$) for algorithms that attempt to select the best sequence of labels by searching through the entire space of possible label combinations, although it can be significantly higher than the running time of O($\sum_{i=1}^{n} N_{w_i}$) for individual data labeling.

2.7 Other Algorithms for Sequence Data Labeling

It is interesting to contrast our algorithm with previously proposed models for sequence data labeling, e.g. Hidden Markov Models, Maximum Entropy Markov Models, or Conditional Random Fields. Although they differ in the model used (generative, discriminative, or dual), and the type of probabilities involved (joint or conditional), these previous algorithms are all parameterized algorithms that typically require parameter training through maximization of likelihood on training examples. In these models, parameters that maximize sequence probabilities are *learned* from a corpus during a *training phase*, and then applied to the annotation of new unseen data. Instead, in the algorithm proposed in this paper, the likelihood of a sequence of labels is determined during *test phase*, through random walks performed on the label graph built for the data to be annotated. While current evaluations of our algorithm are performed on an unsupervised labeling task, future work will consider the evaluation of the algorithm in the presence of an annotated corpus, which will allow for direct comparison with these previously proposed models for sequence data labeling.

3 Experiments in Word Sense Disambiguation

The algorithm for sequence data labeling is illustrated and tested on an all-words word sense disambiguation problem. Word sense disambiguation is a labeling task consisting of assigning the correct meaning to each open-class word in a sequence (usually a sentence). Most of the efforts for solving this problem were concentrated so far toward targeted supervised learning, where each sense tagged occurrence of a particular word is transformed into a feature vector used in an automatic learning process. The applicability of such supervised algorithms is however limited to those few words for which sense tagged data is available, and their accuracy is strongly connected to the amount of labeled data available at hand. Instead, algorithms that attempt to disambiguate all-words in unrestricted text have received significantly less attention, as the development and success of such algorithms has been hindered by both (a) lack of resources (training data), and (b) efficiency aspects resulting from the large size of the problem.

3.1 Graph-based Sequence Data Labeling for Unsupervised Word Sense Disambiguation

To apply the graph-based sequence data labeling algorithm to the disambiguation of an input text, we need information on labels (word senses) and dependencies (word sense dependencies). Word senses can be easily obtained from any sense inventory, e.g. WordNet or LDOCE. Sense dependencies can be derived in various ways, depending on the type of resources available for the language and/or domain at hand. In this paper, we explore the unsupervised derivation of sense dependencies using information drawn from machine readable dictionaries, which is general and can be applied to any language or domain for which a sense inventory is available.

Relying exclusively on a machine readable dictionary, a sense dependency can be defined as a measure of similarity between word senses. There are several metrics that can be used for this purpose, see for instance (Budanitsky and Hirst, 2001) for an overview. However, most of them rely on measures of semantic distance computed on semantic networks, and thus they are limited by the availability of explicitly encoded semantic relations (e.g. *is-a*, *part-of*). To maintain the unsupervised aspect of the algorithm, we chose instead to use a measure of similarity based on sense definitions, which can be computed on any dictionary, and can be evaluated across different parts-of-speech.

Given two word senses and their corresponding definitions, the sense similarity is determined as a function of definition overlap, measured as the number of common tokens between the two definitions, after running them through a simple filter that eliminates all stop-words. To avoid promoting long definitions, we also use a normalization factor, and divide the content overlap of the two definitions with the length of each definition. This sense similarity measure is inspired by the definition of the Lesk algorithm (Lesk, 1986).

Starting with a sense inventory and a function for computing sense dependencies, the application of the sequence data labeling algorithm to the unsupervised disambiguation of a new text proceeds as follows. First, for the given text, a label graph is built by adding a vertex for each possible sense for all open-class words in the text. Next, weighted edges are drawn using the definition-based semantic similarity measure, computed for all pairs of senses for words found within a certain distance ($MaxDist$, as defined in Algorithm 1). Once the graph is constructed, the graph-based ranking algorithm is applied, and a score is determined for all word senses in the graph. Finally, for each open-class word in the text, we select the vertex in the label graph which has the highest score, and label the word with the corresponding word sense.

3.2 An Example

Consider the task of assigning senses to the words in the text *The church bells no longer rung on Sundays*[2]. For the purpose of illustration, we assume at most three senses for each word, which are shown in Figure 2. Word senses and definitions are obtained from the WordNet sense inventory (Miller, 1995). All word senses are added as vertices in the label graph, and weighted edges are drawn as dependencies among word senses, derived using the definition-based similarity measure (no edges are drawn between word senses with a similarity of zero). The resulting label graph is an undirected weighted graph, as shown in Figure 2. After running the ranking algorithm, scores are identified for each word-sense in the graph, indicated between brackets next to each node. Selecting for each word the sense with the largest score results in the following sense assignment: *The church#2 bells#1*

The **church bells** no longer **rung** on **Sundays**.

church
1: one of the groups of Christians who have their own beliefs and forms of worship
2: a place for public (especially Christian) worship
3: a service conducted in a church

bell
1: a hollow device made of metal that makes a ringing sound when struck
2: a push button at an outer door that gives a ringing or buzzing signal when pushed
3: the sound of a bell

ring
1: make a ringing sound
2: ring or echo with sound
3: make (bells) ring, often for the purposes of musical edifi cation

Sunday
1: fi rst day of the week; observed as a day of rest and worship by most Christians

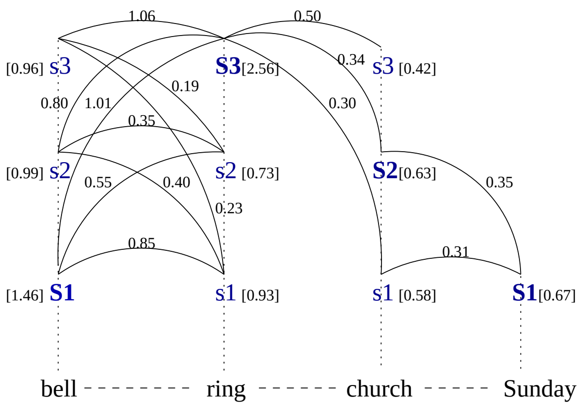


Figure 2: The label graph for assigning senses to words in the sentence *The church bells no longer rung on Sundays.*

[2]Example drawn from the data set provided during the SENSEVAL-2 English all-words task. Manual sense annotations were also made available for this data.

no longer rung#3 on Sundays#1, which is correct according to annotations performed by professional lexicographers.

3.3 Results and Discussion

The algorithm was primarily evaluated on the SENSEVAL-2 English all-words data set, consisting of three documents from Penn Treebank, with 2,456 open-class words (Palmer et al., 2001). Unlike other sense-annotated data sets, e.g. SENSEVAL-3 or SemCor, SENSEVAL-2 is the only testbed for all-words word sense disambiguation that includes a sense map, which allows for additional coarse-grained sense evaluations. Moreover, there is a larger body of previous work that was evaluated on this data set, which can be used as a base of comparison.

The performance of our algorithm is compared with the disambiguation accuracy obtained with a variation of the Lesk algorithm[3] (Lesk, 1986), which selects the meaning of an open-class word by finding the word sense that leads to the highest overlap between the corresponding dictionary definition and the current context. Similar to the definition similarity function used in the graph-based disambiguation algorithm (Section 3.1), the overlap measure used in the Lesk implementation does not take into account stop-words, and it is normalized with the length of each definition to avoid promoting longer definitions.

We are thus comparing the performance of sequence data labeling, which takes into account label dependencies, with individual data labeling, where a label is selected independent of the other labels in the text. Note that both algorithms rely on the same knowledge source, i.e. dictionary definitions, and thus they are directly comparable. Moreover, none of the algorithms take into account the dictionary sense order (e.g. the most frequent sense provided by WordNet), and therefore they are both fully unsupervised.

Table 1 shows precision and recall figures[4] for a context size ($MaxDist$) equal to the length of each sentence, using: (a) sequence data labeling with iterative graph-based algorithms; (b) individual data labeling with a version of the Lesk algorithm; (c) random baseline. Evaluations are run for both fine-grained and coarse-grained sense distinctions, to determine the algorithm performance under different classification granularities.

The accuracy of the graph-based sequence data labeling algorithm exceeds by a large margin the individual data labeling algorithm, resulting in 10.7% error rate reduction for fine-grained sense distinctions, which is statistically significant ($p < 0.0001$, paired t-test). Performance improvements are equally distributed across all parts-of-speech, with comparable improvements obtained for nouns, verbs, and adjectives. A similar error rate reduction of 11.0% is obtained for coarse-grained sense distinctions, which suggests that the performance of the graph-based sequence data labeling algorithm does not depend on classification granularity, and similar improvements over individual data labeling can be obtained regardless of the average number of labels per word.

We also measured the variation of performance with context size, and evaluated the disambiguation accuracy for both algorithms for a window size ranging from two words to an entire sentence. The window size parameter limits the number of surrounding words considered when seeking label dependencies (sequence data labeling), or the words counted in the measure of definition–context overlap (individual data labeling). Figure 3 plots the disambiguation accuracy of the two algorithms as a function of context size. As seen in the figure, both algorithms benefit from larger contexts, with a steady increase in performance observed for increasingly larger window sizes. Although the initial growth observed for the sequence data labeling algorithm is somewhat sharper, the gap between the two curves stabilizes for window sizes larger than five words, which suggests that the improvement in performance achieved with sequence data labeling over individual data labeling does not depend on the size of available context.

The algorithm was also evaluated on two other data sets, SENSEVAL-3 English all-words data (Snyder and Palmer, 2004) and a subset of SemCor (Miller et al., 1993), although only fine-grained sense evaluations could be conducted on these test sets. The disambiguation precision on the SENSEVAL-3 data was measured at 52.2% using sequence data labeling, compared to 48.1% obtained with individual

[3]Given a sequence of words, the original Lesk algorithm attempts to identify the combination of word senses that maximizes the redundancy (overlap) across all corresponding definitions. The algorithm was later improved through a method for simulated annealing (Cowie et al., 1992), which solved the combinatorial explosion of word senses, while still finding an optimal solution. However, recent comparative evaluations of different variants of the Lesk algorithm have shown that the performance of the original algorithm is significantly exceeded by an algorithm variation that relies on the overlap between word senses and current context (Vasilescu et al., 2004). We are thus using this latter Lesk variant in our implementation.

[4]Recall is particularly low for each individual part-of-speech because it is calculated with respect to the entire data set. The overall precision and recall figures coincide, reflecting the 100% coverage of the algorithm.

Part-of speech	Fine-grained sense distinctions						Coarse-grained sense distinctions					
	Random baseline		Individual (Lesk)		Sequence (graph-based)		Random baseline		Individual (Lesk)		Sequence (graph-based)	
	P	R	P	R	P	R	P	R	P	R	P	R
Noun	41.4%	19.4%	50.3%	23.6%	57.5%	27.0%	42.7%	20.0%	51.4%	24.1%	58.8%	27.5%
Verb	20.7%	3.9%	30.5%	5.7%	36.5%	6.9%	22.8%	4.3%	31.9%	6.0%	37.9%	7.1%
Adjective	41.3%	9.3%	49.1%	11.0%	56.7%	12.7%	42.6%	42.6%	49.8%	11.2%	57.6%	12.9%
Adverb	44.6%	5.2%	64.6%	7.6%	70.9%	8.3%	40.7%	4.8%	65.3%	7.7%	71.9%	8.5%
ALL	37.9%	37.9%	48.7%	48.7%	**54.2%**	**54.2%**	38.7%	38.7%	49.8%	49.8%	**55.3%**	**55.3%**

Table 1: Precision and recall for graph-based sequence data labeling, individual data labeling, and random baseline, for fine-grained and coarse-grained sense distinctions.

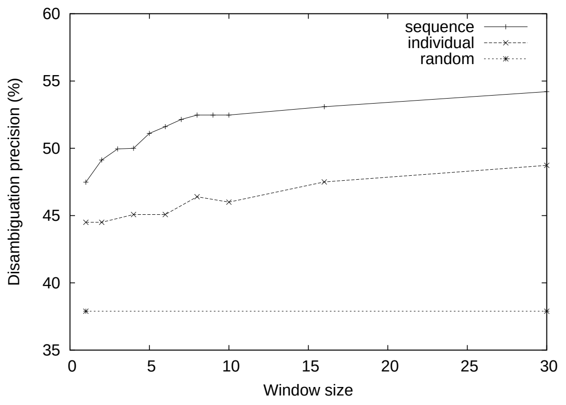


Figure 3: Disambiguation results using sequence data labeling, individual labeling, and random baseline, for various context sizes.

data labeling, and 34.3% achieved through random sense assignment. The average disambiguation figure obtained on all the words in a random subset of 10 SemCor documents, covering different domains, was 56.5% for sequence data labeling, 47.4% for individual labeling, and 35.3% for the random baseline.

Comparison with Related Work

For a given sequence of ambiguous words, the original definition of the Lesk algorithm (Lesk, 1986), and more recent improvements based on simulated annealing (Cowie et al., 1992), seek to identify the combination of senses that maximizes the overlap among their dictionary definitions. Tests performed with this algorithm on the SENSEVAL-2 data set resulted in a disambiguation accuracy of 39.5%. This precision is exceeded by the Lesk algorithm variation used in the experiments reported in this paper, which measures the overlap between sense definitions and the current context, for a precision of 48.7% on the same data set (see Table 1). In the SENSEVAL-2 evaluations, the best performing fully unsupervised algorithm[5] was developed by (Litkowski, 2001), who combines analysis of multiword units and contextual clues based on collocations and content words from dictionary definitions and examples, for an overall precision and recall of 45.1%. More recently, (McCarthy et al., 2004) reports one of the best results on the SENSEVAL-2 data set, using an algorithm that automatically derives the most frequent sense for a word using distributional similarities learned from a large raw corpus, for a disambiguation precision of 53.0% and a recall of 49.0%.

Another related line of work consists of the disambiguation algorithms based on lexical chains (Morris and Hirst, 1991), and the more recent improvements reported in (Galley and McKeown, 2003) – where threads of meaning are identified throughout a text. Lexical chains however only take into account connections between concepts identified in a static way, without considering the importance of the concepts that participate in a relation, which is recursively determined in our algorithm. Moreover, the construction of lexical chains requires structured dictionaries such as WordNet, with explicitly defined semantic relations between word senses, whereas our algorithm can also work with simple unstructured dictionaries that provide only word sense definitions. (Galley and McKeown, 2003) evaluated their algorithm on the nouns from a subset of SEMCOR, reporting 62.09% disambiguation precision. The performance of our algorithm on the same subset of SEMCOR nouns was measured at 64.2%[6]. Finally, another disambiguation method relying on graph algorithms that exploit the

[5]Algorithms that integrate the most frequent sense in WordNet are not considered here, since this represents a supervised knowledge source (WordNet sense frequencies are derived from a sense-annotated corpus).

[6]Note that the results are not directly comparable, since (Galley and McKeown, 2003) used the WordNet sense order to break the ties, whereas we assume that such sense order frequency is not available, and thus we break the ties through random choice.

structure of semantic networks was proposed in (Mihalcea et al., 2004), with a disambiguation accuracy of 50.9% measured on all the words in the SENSEVAL-2 data set.

Although it relies exclusively on dictionary definitions, the graph-based sequence data labeling algorithm proposed in this paper, with its overall performance of 54.2%, exceeds significantly the accuracy of all these previously proposed unsupervised word sense disambiguation methods, proving the benefits of taking into account label dependencies when annotating sequence data. An additional interesting benefit of the algorithm is that it provides a ranking over word senses, and thus the selection of two or more most probable senses for each word is also possible.

4 Conclusions

We proposed a graphical algorithm for sequence data labeling that relies on random walks on graphs encoding label dependencies. Through the label graphs it builds for a given sequence of words, the algorithm exploits relations between word labels, and implements a concept of *recommendation*. A label recommends other related labels, and the strength of the recommendation is recursively computed based on the importance of the labels making the recommendation. In this way, the algorithm simultaneously annotates all the words in an input sequence, by identifying the most probable (most *recommended*) set of labels.

The algorithm was illustrated and tested on an unsupervised word sense disambiguation problem, targeting the annotation of all words in unrestricted texts. Through experiments performed on standard sense-annotated data sets, the graph-based sequence data labeling algorithm was shown to significantly outperform the accuracy achieved through individual data labeling, resulting in a statistically significant error rate reduction of 10.7%. The disambiguation method was also shown to exceed the performance of previously proposed unsupervised word sense disambiguation algorithms. Moreover, comparative results obtained under various experimental settings have shown that the algorithm is robust to changes in classification granularity and context size.

Acknowledgments
This work was partially supported by a National Science Foundation grant IIS-0336793.

References

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

A. Budanitsky and G. Hirst. 2001. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources*, Pittsburgh.

J. Cowie, L. Guthrie, and J. Guthrie. 1992. Lexical disambiguation using simulated annealing. In *Proceedings of the 5th International Conference on Computational Linguistics (COLING 1992)*.

D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing ANLP-92*.

M. Galley and K. McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, Acapulco, Mexico, August.

G. Grimmett and D. Stirzaker. 1989. *Probability and Random Processes*. Oxford University Press.

M.E. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference 1986*, Toronto.

K. Litkowski. 2001. Use of machine readable dictionaries in word sense disambiguation for Senseval-2. In *Proceedings of ACL/SIGLEX Senseval-2*, Toulouse, France.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. Using automatically acquired predominant senses for word sense disambiguation. In *Proceedings of ACL/SIGLEX Senseval-3*, Barcelona, Spain.

R. Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20st International Conference on Computational Linguistics (COLING 2004)*.

G. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, Plainsboro, New Jersey.

G. Miller. 1995. Wordnet: A lexical database. *Communication of the ACM*, 38(11):39–41.

J. Morris and G. Hirst. 1991. Lexical cohesion, the thesaurus, and the structure of text. *Computational Linguistics*, 17(1):21–48.

M. Palmer, C. Fellbaum, S. Cotton, L. Delfs, and H.T. Dang. 2001. English tasks: all-words and verb lexical sample. In *Proceedings of ACL/SIGLEX Senseval-2*, Toulouse, France.

B. Snyder and M. Palmer. 2004. The English all-words task. In *Proceedings of ACL/SIGLEX Senseval-3*, Barcelona, Spain.

F. Vasilescu, P. Langlais, and G. Lapalme. 2004. Evaluating variants of the Lesk approach for disambiguating words. In *Proceedings of the Conference of Language Resources and Evaluations (LREC 2004)*.

Domain-Specific Sense Distributions and Predominant Sense Acquisition

Rob Koeling & Diana McCarthy & John Carroll
Department of Informatics,
University of Sussex
Brighton BN1 9QH, UK
{*robk,dianam,johnca*}*@sussex.ac.uk*

Abstract

Distributions of the senses of words are often highly skewed. This fact is exploited by word sense disambiguation (WSD) systems which back off to the predominant sense of a word when contextual clues are not strong enough. The domain of a document has a strong influence on the sense distribution of words, but it is not feasible to produce large manually annotated corpora for every domain of interest. In this paper we describe the construction of three sense annotated corpora in different domains for a sample of English words. We apply an existing method for acquiring predominant sense information automatically from raw text, and for our sample demonstrate that (1) acquiring such information automatically from a mixed-domain corpus is more accurate than deriving it from SemCor, and (2) acquiring it automatically from text in the same domain as the target domain performs best by a large margin. We also show that for an all words WSD task this automatic method is best focussed on words that are salient to the domain, and on words with a different acquired predominant sense in that domain compared to that acquired from a balanced corpus.

1 Introduction

From analysis of manually sense tagged corpora, Kilgarriff (2004) has demonstrated that distributions of the senses of words are often highly skewed. Most researchers working on word sense disambiguation (WSD) use manually sense tagged data such as SemCor (Miller et al., 1993) to train statistical classifiers, but also use the information in SemCor on the overall sense distribution for each word as a back-off model. In WSD, the heuristic of just choosing the most frequent sense of a word is very powerful, especially for words with highly skewed sense distributions (Yarowsky and Florian, 2002). Indeed, only 5 out of the 26 systems in the recent SENSEVAL-3 English all words task (Snyder and Palmer, 2004) outperformed the heuristic of choosing the most frequent sense as derived from SemCor (which would give 61.5% precision and recall[1]). Furthermore, systems that did outperform the first sense heuristic did so only by a small margin (the top score being 65% precision and recall).

Over a decade ago, Gale et al. (1992) observed the tendency for one sense of a word to prevail in a given discourse. To take advantage of this, a method for automatically determining the "one sense" given a discourse or document is required. Magnini et al. (2002) have shown that information about the domain of a document is very useful for WSD. This is because many concepts are specific to particular domains, and for many words their most likely meaning in context is strongly correlated to the domain of the document they appear in. Thus, since word sense distributions are skewed and depend on the domain at hand we would like to know *for each domain of application* the most likely sense of a word.

However, there are no extant *domain-specific* sense tagged corpora to derive such sense distribution information from. Producing them would be extremely costly, since a substantial corpus would have to be annotated by hand for every domain of interest. In response to this problem, McCarthy et al. (2004) proposed a method for *automatically* inducing the

[1]This figure is the mean of two different estimates (Snyder and Palmer, 2004), the difference being due to multiword handling.

predominant sense of a word from raw text. They carried out a limited test of their method on text in two domains using subject field codes (Magnini and Cavaglià, 2000) to assess whether the acquired predominant sense information was broadly consistent with the domain of the text it was acquired from. But they did not evaluate their method on hand-tagged domain-specific corpora since there was no such data publicly available.

In this paper, we evaluate the method on domain specific text by creating a sense-annotated gold standard[2] for a sample of words. We used a lexical sample because the cost of hand tagging several corpora for an all-words task would be prohibitive. We show that the sense distributions of words in this lexical sample differ depending on domain. We also show that sense distributions are more skewed in domain-specific text. Using McCarthy et al.'s method, we automatically acquire predominant sense information for the lexical sample from the (raw) corpora, and evaluate the accuracy of this and predominant sense information derived from SemCor. We show that in our domains and for these words, first sense information automatically acquired from a general corpus is more accurate than first senses derived from SemCor. We also show that deriving first sense information from text in the same domain as the target data performs best, particularly when focusing on words which are salient to that domain.

The paper is structured as follows. In section 2 we summarise McCarthy et al.'s predominant sense method. We then (section 3) describe the new gold standard corpora, and evaluate predominant sense accuracy (section 4). We discuss the results with a proposal for applying the method to an all-words task, and an analysis of our results in terms of this proposal before concluding with future directions.

2 Finding Predominant Senses

We use the method described in McCarthy et al. (2004) for finding predominant senses from raw text. The method uses a thesaurus obtained from the text by parsing, extracting grammatical relations and then listing each word (w) with its top k nearest neighbours, where k is a constant. Like McCarthy et al. (2004) we use $k = 50$ and obtain our thesaurus using the distributional similarity metric described by Lin (1998). We use WordNet (WN) as our sense inventory. The senses of a word w are each assigned a ranking score which sums over the distributional similarity scores of the neighbours and weights each neighbour's score by a WN Similarity score (Patwardhan and Pedersen, 2003) between the sense of w and the sense of the neighbour that maximises the WN Similarity score. This weight is normalised by the sum of such WN similarity scores between all senses of w and and the senses of the neighbour that maximises this score. We use the WN Similarity **jcn** score (Jiang and Conrath, 1997) since this gave reasonable results for McCarthy et al. and it is efficient at run time given precompilation of frequency information. The **jcn** measure needs word frequency information, which we obtained from the British National Corpus (BNC) (Leech, 1992). The distributional thesaurus was constructed using subject, direct object adjective modifier and noun modifier relations.

3 Creating the Three Gold Standards

In our experiments, we compare for a sample of nouns the sense rankings created from a balanced corpus (the BNC) with rankings created from domain-specific corpora (FINANCE and SPORTS) extracted from the Reuters corpus (Rose et al., 2002). In more detail, the three corpora are:

BNC: The 'written' documents, amounting to 3209 documents (around 89.7M words), and covering a wide range of topic domains.

FINANCE: 117734 FINANCE documents (around 32.5M words) topic codes: ECAT and MCAT

SPORTS: 35317 SPORTS documents (around 9.1M words) topic code: GSPO

We computed thesauruses for each of these corpora using the procedure outlined in section 2.

3.1 Word Selection

In our experiments we used FINANCE and SPORTS domains. To ensure that a significant number of the chosen words are relevant for these domains, we did not choose the words for our experiments completely randomly. The first selection criterion we applied used the Subject Field Code (SFC) re-

[2]This resource will be made publicly available for research purposes in the near future.

source (Magnini and Cavaglià, 2000), which assigns domain labels to synsets in WN version 1.6. We selected all the polysemous nouns in WN 1.6 that have at least one synset labelled SPORT and one synset labelled FINANCE. This reduced the set of words to 38. However, some of these words were fairly obscure, did not occur frequently enough in one of the domain corpora or were simply too polysemous. We narrowed down the set of words using the criteria: (1) frequency in the BNC $\geq$ 1000, (2) at most 12 senses, and (3) at least 75 examples in each corpus. Finally a couple of words were removed because the domain-specific sense was particularly obscure[3]. The resulting set consists of 17 words[4]: *club, manager, record, right, bill, check, competition, conversion, crew, delivery, division, fishing, reserve, return, score, receiver, running*

We refer to this set of words as **F&S cds**. The first four words occur in the BNC with high frequency ($\geq$ 10000 occurrences), the last two with low frequency ($\leq$ 2000) and the rest are mid-frequency.

Three further sets of words were selected on the basis of domain salience. We chose eight words that are particularly salient in the Sport corpus (referred to as **S sal**), eight in the Finance corpus (**F sal**), and seven that had equal (not necessarily high) salience in both, (**eq sal**). We computed salience as a ratio of normalised document frequencies, using the formula

$$S(w, d) = \frac{N_{wd}/N_d}{N_w/N}$$

where N_{wd} is the number of documents in domain d containing the noun (lemma) w, N_d is the number of documents in domain d, N_w is the total number of documents containing the noun w and N is the total number of documents.

To obtain the sets **S sal**, **F sal** and **eq sal** we generated the 50 most salient words for both domains and 50 words that were equally salient for both domains. These lists of 50 words were subjected to the same constraints as set **F&S cds**, that is occurring in the BNC $\geq$ 1000, having at most 12 senses, and having at least 75 examples in each corpus. From the remaining words we randomly sampled 8 words from the **Sport** salience list and **Finance** list and 7 from the salience list for words with equal salience in both domains. The resulting sets of words are:

S sal: *fan, star, transfer, striker, goal, title, tie, coach*

F sal: *package, chip, bond, market, strike, bank, share, target*

eq sal: *will, phase, half, top, performance, level, country*

The average degree of polysemy for this set of 40 nouns in WN (version 1.7.1) is 6.6.

3.2 The Annotation Task

For the annotation task we recruited linguistics students from two universities. All ten annotators are native speakers of English.

We set up annotation as an Open Mind Word Expert task[5]. Open Mind is a web based system for annotating sentences. The user can choose a word from a pull down menu. When a word is selected, the user is presented with a list of sense definitions. The sense definitions were taken from WN1.7.1 and presented in random order. Below the sense definitions, sentences with the target word (highlighted) are given. Left of the sentence on the screen, there are as many tick-boxes as there are senses for the word plus boxes for 'unclear' and 'unlisted-sense'. The annotator is expected to first read the sense definitions carefully and then, after reading the sentence, decide which sense is best for the instance of the word in a particular sentence. Only the sentence in which the word appears is presented (not more surrounding sentences). In case the sentence does not give enough evidence to decide, the annotator is expected to check the 'unclear' box. When the correct sense is not listed, the annotator should check the 'unlisted-sense' box.

The sentences to be annotated were randomly sampled from the corpora. The corpora were first part of speech tagged and lemmatised using RASP (Briscoe and Carroll, 2002). Up to 125 sentences were randomly selected for each word from each corpus. Sentences with clear problems (e.g. containing a begin or end of document marker, or mostly not text) were removed. The first 100 remaining sentences were selected for the task. For a few

[3]For example the Finance sense of 'eagle' (a former gold coin in US worth 10 dollars) is very unlikely to be found.

[4]One more word, 'pitch', was in the original selection. However, we did not obtain enough usable annotated sentences (section 3.2) for this particular word and therefore it was discarded.

[5]http://www.teach-computers.org/word-expert/english/

words there were not exactly 100 sentences per corpus available. The Reuters corpus contains quite a few duplicate documents. No attempts were made to remove duplicates.

3.3 Characterisation of the Annotated Data

Most of the sentences were annotated by at least three people. Some sentences were only done by two annotators. The complete set of data comprises 33225 tagging acts.

The inter-annotator agreement on the complete set of data was 65%[6]. For the BNC data it was 60%, for the Sports data 65% and for the Finance data 69%. This is lower than reported for other sets of annotated data (for example it was 75% for the nouns in the SENSEVAL-2 English all-words task), but quite close to the reported 62.8% agreement between the first two taggings for single noun tagging for the SENSEVAL-3 English lexical sample task (Mihalcea et al., 2004). The fairest comparison is probably between the latter and the inter-annotator agreement for the BNC data. Reasons why our agreement is relatively low include the fact that almost all of the sentences are annotated by three people, and also the high degree of polysemy of this set of words.

Problematic cases

The unlisted category was used as a miscellaneous category. In some cases a sense was truly missing from the inventory (e.g. the word ‘tie’ has a ‘game’ sense in British English which is not included in WN 1.7.1). In other cases we had not recognised that the word was really part of a multiword (e.g. a number of sentences for the word ‘chip’ contained the multiword ‘blue chip’). Finally there were a number of cases where the word had been assigned the wrong part of speech tag (e.g. the verb ‘will’ had often been mistagged as a noun). We identified and removed all these systematic problem cases from the unlisted senses. After removing the problematic unlisted cases, we had between 0.9% (FINANCE) and 4.5% (SPORTS) unlisted instances left. We also had between 1.8% (SPORTS) and 4.8% (BNC) unclear instances. The percentage of unlisted instances reflects the fit of WN to the data whilst that of unclear cases reflects the generality of the corpus.

[6]To compute inter-annotator agreement we used Amruta Purandare and Ted Pedersen’s OMtoSVAL2 Package version 0.01.

The sense distributions

WSD accuracy is strongly related to the entropy of the sense distribution of the target word (Yarowsky and Florian, 2002). The more skewed the sense distribution is towards a small percentage of the senses, the lower the entropy. Accuracy is related to this because there is more data (both training and test) shared between fewer of the senses. When the first sense is very predominant (exceeding 80%) it is hard for any WSD system to beat the heuristic of always selecting that sense (Yarowsky and Florian, 2002).

The sense distribution for a given word may vary depending on the domain of the text being processed. In some cases, this may result in a different predominant sense; other characteristics of the sense distribution may also differ such as entropy of the sense distribution and the dominance of the predominant sense. In Table 1 we show the entropy per word in our sample and relative frequency (relfr) of its first sense (fs), for each of our three gold standard annotated corpora. We compute the entropy of a word’s sense distribution as a fraction of the possible entropy (Yarowsky and Florian, 2002)

$$H_r(P) = \frac{H(P)}{log_2(\#senses)}$$

where $H(P) = -\sum_{i \in senses} p(i) log_2 p(i)$. This measure reduces the impact of the number of senses of a word and focuses on the uncertainty within the distribution. For each corpus, we also show the average entropy and average relative frequency of the first sense over all words.

From Table 1 we can see that for the vast majority of words the entropy is highest in the BNC. However there are exceptions: *return*, *fan* and *title* for FINANCE and *return*, *half*, *level*, *running strike* and *share* for SPORTS. Surprisingly, **eq sal** words, which are not particularly salient in either domain, also typically have lower entropy in the domain specific corpora compared to the BNC. Presumably this is simply because of this small set of words, which seem particularly skewed to the financial domain. Note that whilst the distributions in the domain-specific corpora are more skewed towards a predominant sense, only 7 of the 40 words in the FINANCE corpus and 5 of the 40 words in the SPORTS corpus have only one sense attested. Thus, even in domain-specific corpora ambiguity is

Training	Testing		
	BNC	FINANCE	SPORTS
BNC	**40.7**	43.3	33.2
FINANCE	39.1	**49.9**	24.0
SPORTS	25.7	19.7	**43.7**
Random BL	19.8	19.6	19.4
SemCor FS	32.0 (32.9)	33.9 (35.0)	16.3 (16.8)

Table 2: WSD using predominant senses, training and testing on all domain combinations.

Test - Train	**F&S cds**	**F sal**	**S sal**	**eq sal**
BNC-APPR	33.3	51.5	39.7	48.0
BNC-SC	28.3	44.0	24.6	36.2
FINANCE-APPR	37.0	70.2	38.5	70.1
FINANCE-SC	30.3	51.1	22.9	33.5
SPORTS-APPR	42.6	18.1	65.7	46.9
SPORTS-SC	9.4	38.1	13.2	12.2

Table 3: WSD using predominant senses, with training data from the same domain or from SemCor.

still present, even though it is less than for general text. We show the sense number of the first sense (fs) alongside the relative frequency of that sense. We use 'ucl' for unclear and 'unl' for unlisted senses where these are predominant in our annotated data. Although the predominant sense of a word is not always the domain-specific sense in a domain-specific corpus, the domain-specific senses typically occur more than they do in non-relevant corpora. For example, sense 11 of *return* (a tennis stroke) was not the first sense in SPORTS, however it did have a relative frequency of 19% in that corpus and was absent from BNC and FINANCE.

4 Predominant Sense Evaluation

We have run the predominant sense finding algorithm on the raw text of each of the three corpora in turn (the first step being to compute a distributional similarity thesaurus for each, as outlined in section 2). We evaluate the accuracy of performing WSD purely with the predominant sense heuristic using all 9 combinations of training and test corpora. The results are presented in Table 2. The random baseline is $\sum_{i \in tokens} \frac{1}{\#senses(i)}$. We also give the accuracy using a first sense heuristic from SemCor ('SemCor FS'); the precision is given alongside in brackets because a predominant sense is not supplied by SemCor for every word. [7] The automatic method proposes a predominant sense in every case.

The best results are obtained when training on a domain relevant corpus. In all cases, when training on appropriate training data the automatic method for finding predominant senses beats both the random baseline and the baseline provided by SemCor.

Table 3 compares WSD accuracy using the automatically acquired first sense on the 4 categories of words **F&S cds**, **F sal**, **S sal** and **eq sal** separately. Results using the training data from the appropriate domain (e.g. SPORTS training data for SPORTS test data) are indicated with 'APPR' and contrasted with the results using SemCor data, indicated with 'SC'. [8] We see that for words which are pertinent to the domain of the test text, it pays to use domain specific training data. In some other cases, e.g. **F sal** tested on SPORTS, it is better to use SemCor data. For the **eq sal** words, accuracy is highest when FINANCE data is used for training, reflecting their bias to financial senses as noted in section 3.3.

5 Discussion

We are not aware of any other domain-specific manually sense tagged corpora. We have created sense tagged corpora from two specific domains for a sample of words, and a similar resource from a balanced corpus which covers a wide range of domains. We have used these resources to do a quantitative evaluation which demonstrates that automatic acquisition of predominant senses outperforms the SemCor baseline for this sample of words.

The domain-specific manually sense tagged resource is an interesting source of information in itself. It shows for example that (at least for this particular lexical sample), the predominant sense is much more dominant in a specific domain than it is in the general case, even for words which are not particularly salient in that domain. Similar observations can be made about the average number of encountered senses and the skew of the sense distributions. It also shows that although the predominant sense is more dominant and domain-specific

[7]There is one such word in our sample, *striker*.

[8]For SemCor, precision figures for the **S sal** words are up to 4% higher than the accuracy figures given, however they are still lower than accuracy using the domain specific corpora; we leave them out due to lack of space.

senses are used more within a specific domain, there is still a need for taking local context into account when disambiguating words. The predominant sense heuristic is hard to beat for some words within a domain, but others remain highly ambiguous even within a specific domain. The *return* example in section 3.3 illustrates this.

Our results are for a lexical sample because we did not have the resources to produce manually tagged domain-specific corpora for an all words task. Although sense distribution data derived from SemCor can be more accurate than such information derived automatically (McCarthy et al., 2004), in a given domain there will be words for which the SemCor frequency distributions are inappropriate or unavailable. The work presented here demonstrates that the automatic method for finding predominant senses outperforms SemCor on a sample of words, particularly on ones that are salient to a domain. As well as domain-salient words, there will be words which are not particularly salient but still have different distributions than in SemCor. We therefore propose that automatic methods for determining the first sense should be used when either there is no manually tagged data, or the manually tagged data seems to be inappropriate for the word and domain under consideration. While it is trivial to find the words which are absent or infrequent in training data, such as SemCor, it is less obvious how to find words where the training data is not appropriate. One way of finding these words would be to look for differences in the automatic sense rankings of words in domain specific corpora compared to those of the same words in balanced corpora, such as the BNC. We assume that the sense rankings from a balanced text will more or less correlate with a balanced resource such as SemCor. Of course there will be differences in the corpus data, but these will be less radical than those between SemCor and a domain specific corpus. Then the automatic ranking method should be applied in cases where there is a clear deviation in the ranking induced from the domain specific corpus compared to that from the balanced corpus. Otherwise, SemCor is probably more reliable if data for the given word is available.

There are several possibilities for the definition of "clear deviation" above. One could look at differences in the ranking over all words, using a measure such as pairwise agreement of rankings or a ranking correlation coefficient, such as Spearman's. One could also use the rankings to estimate probability distributions and compare the distributions with measures such as alpha-skew divergence (Lee, 1999). A simple definition would be where the rankings assign different predominant senses to a word. Taking this simple definition of deviation, we demonstrate how this might be done for our corpora.

Training	Testing	
	FINANCE	SPORTS
Finance	35.5	-
Sports	-	40.9
SemCor	14.2 (15.3)	10.0

Table 4: WSD accuracy for words with a different first sense to the BNC.

We compared the automatic rankings from the BNC with those from each domain specific corpus (SPORTS and FINANCE) for all polysemous nouns in SemCor. Although the majority are assigned the same first sense in the BNC as in the domain specific corpora, a significant proportion (31% SPORTS and 34% FINANCE) are not. For all words WSD in either of these domains, it would be these words for which automatic ranking should be used. Table 4 shows the WSD accuracy using this approach for the words in our lexical sample with a different automatically computed first sense in the BNC compared to the target domain (SPORTS or FINANCE). We trained on the appropriate domain for each test corpus, and compared this with using SemCor first sense data. The results show clearly that using this approach to decide whether to use automatic sense rankings performs much better than always using SemCor rankings.

6 Conclusions

The method for automatically finding the predominant sense beat SemCor consistently in our experiments. So for some words, it pays to obtain automatic information on frequency distributions from appropriate corpora. Our sense annotated corpora exhibit higher entropy for word sense distributions for domain-specific text, even for words which are not specific to that domain. They also show that different senses predominate in different domains and that dominance of the first sense varies to a great extent, depending on the word. Previous work in all words WSD has indicated that techniques using hand-tagged resources outperform unsupervised methods. However, we demonstrate that it is possible to apply a fully automatic method to a subset of pertinent words to improve WSD accuracy. The automatic method seems to lead to better performance for words that are salient to a domain. There are also other words which though not particularly domain-salient, have a different sense distribution to that anticipated for a balanced corpus. We propose that in order to tackle an all words task, automatic methods should be applied to words which have a substantial difference in sense ranking compared to that obtained from a balanced corpus. We demonstrate that for a set of words which meet this condition, the performance of the automatic method is far better than when using data from SemCor. We will do further work to ascertain the best method for quantifying "substantial change".

We also intend to exploit the automatic ranking to obtain information on sense frequency distributions (rather than just predominant senses) given the genre as well as the domain of the text. We plan to combine this with local context, using collocates of neighbours in the thesaurus, for contextual WSD.

Acknowledgements

We would like to thank Siddharth Patwardhan and Ted Pedersen for making the WN Similarity package available, Rada Mihalcea and Tim Chklovski for making the Open Mind software available to us and Julie Weeds for the thesaurus software. The work was funded by EU-2001-34460 project MEANING, UK EPSRC project "Ranking Word Sense for Word Sense Disambiguation" and the UK Royal Society.

References

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of LREC-2002*, pages 1499–1504, Las Palmas de Gran Canaria.

William Gale, Kenneth Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pages 233–237.

Jay Jiang and David Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference on Research in Computational Linguistics*, Taiwan.

Adam Kilgarriff. 2004. How dominant is the commonest sense of a word? In *Proceedings of Text, Speech, Dialogue*, Brno, Czech Republic.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Geoffrey Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL 98*, Montreal, Canada.

Bernardo Magnini and Gabriela Cavaglià. 2000. Integrating subject field codes into WordNet. In *Proceedings of LREC-2000*, Athens, Greece.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 280–287, Barcelona, Spain.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgariff. 2004. The SENSEVAL-3 English lexical sample task. In *Proceedings of the* SENSEVAL-3 *workshop*, pages 25–28.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 303–308. Morgan Kaufman.

Siddharth Patwardhan and Ted Pedersen. 2003. The cpan wordnet::similarity package. http://search.cpan.org/šid/WordNet-Similarity/.

Tony G. Rose, Mark Stevenson, and Miles Whitehead. 2002. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of LREC-2002*, Las Palmas de Gran Canaria.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3*, pages 41–43, Barcelona, Spain.

David Yarowsky and Radu Florian. 2002. Evaluating sense disambiguation performance across diverse parameter spaces. *Natural Language Engineering*, 8(4):293–310.

	BNC		FINANCE		SPORTS	
word	$H_r(P)$	relf (fs)	$H_r(P)$	relf (fs)	$H_r(P)$	relf (fs)
			F&S cds			
bill	0.503	42.6 (1)	0.284	77.0 (1)	0.478	45.2 (2)
check	0.672	34.4 (6)	0.412	44.2 (1)	0.519	50.0 (1)
club	0.442	75.3 (2)	0.087	96.6 (2)	0.204	90.6 (2)
competition	0.833	42.0 (1)	0.159	95.7 (1)	0.142	95.8 (2)
conversion	0.670	53.2 (9)	0.350	75.6 (8)	0.000	100 (3)
crew	0.726	61.6 (1)	0.343	85.4 (1)	0.508	79.2 (4)
delivery	0.478	74.5 (1)	0.396	72.4 (unc)	0.051	98.0 (6)
division	0.730	34.2 (2)	0.323	76.9 (2)	0.000	100 (7)
fishing	0.922	66.3 (1)	0.500	89.0 (2)	0.422	91.4 (1)
manager	0.839	73.2 (1)	0.252	95.8 (1)	0.420	91.5 (2)
receiver	0.781	47.4 (3)	0.283	89.4 (2)	0.206	92.0 (5)
record	0.779	36.0 (3)	0.287	81.6 (3)	0.422	68.5 (3)
reserve	0.685	50.0 (5)	0.000	100 (2)	0.265	86.4 (3)
return	0.631	33.0 (5)	0.679	34.8 (6)	0.669	28.6 (2 5)
right	0.635	38.6 (1 3)	0.357	71.6 (1)	0.468	60.3 (3)
running	0.621	64.3 (4)	0.485	56.1 (4)	0.955	28.3 (unl)
score	0.682	38.8 (3)	0.476	69.0 (4)	0.200	84.1 (3)
F&S cds averages	0.684	50.9	0.334	77.1	0.349	75.9
			F sal			
bank	0.427	71.3 (1)	0.000	100 (1)	0.247	85.4 (1)
bond	0.499	46.7 (2)	0.000	100 (2)	0.319	75.0 (2)
chip	0.276	82.8 (7)	0.137	92.7 (7)	0.178	91.5 (8)
market	0.751	62.3 (1)	0.524	70.3 (2)	0.734	46.7 (2)
package	0.890	50.0 (1)	0.285	91.8 (1)	0.192	94.6 (1)
share	0.545	62.9 (1)	0.519	65.3 (1)	0.608	47.9 (3)
strike	0.152	93.5 (1)	0.000	100 (1)	0.409	66.7 (unl)
target	0.712	61.6 (5)	0.129	95.6 (5)	0.300	85.4 (5)
F sal averages	0.532	66.4	0.199	89.5	0.373	74.1
			S sal			
coach	0.777	45.7 (1)	0.623	62.5 (5)	0.063	97.9 (1)
fan	0.948	47.6 (3)	0.992	39.5 (3)	0.181	95.0 (2)
goal	0.681	46.9 (2)	0.000	100 (1)	0.245	91.8 (2)
star	0.779	47.7 (6)	0.631	41.7 (2)	0.285	80.9 (2)
striker	0.179	94.0 (1)	0.000	100 (3)	0.000	100 (1)
tie	0.481	45.1 (1)	0.025	99.0 (2)	0.353	51.0 (unl)
title	0.489	50.0 (4)	0.661	42.1 (6)	0.000	100 (4)
transfer	0.600	45.7 (1)	0.316	84.9 (6)	0.168	92.5 (6)
S sal averages	0.617	52.8	0.406	71.2	0.162	88.6
			eq sal			
country	0.729	45.2 (2)	0.195	92.9 (2)	0.459	73.8 (2)
half	0.642	83.7 (1)	0.000	100 (1)	0.798	75.8 (2)
level	0.609	56.0 (1)	0.157	91.5 (1)	0.675	31.1 (unl)
performance	0.987	23.7 (4 5)	0.259	90.1 (2)	0.222	92.0 (5)
phase	0.396	84.7 (2)	0.000	100 (2)	0.000	100 (2)
top	0.593	51.7 (1)	0.035	98.4 (5)	0.063	96.6 (5)
will	0.890	46.9 (2)	0.199	94.3 (2)	0.692	62.2 (2)
eq sal averages	0.692	56.0	0.121	95.3	0.416	75.9
Overall averages	0.642	55.3	0.284	81.6	0.328	78.1

Table 1: Entropy and relative frequency of the first sense in the three gold standards.

Chinese Named Entity Recognition Based on Multiple Features

Youzheng Wu, Jun Zhao, Bo Xu
National Laboratory of Pattern Recognition
Institute of Automation, CAS
Beijing, 100080, China
`(yzwu,jzhao,bxu)@nlpr.ia.ac.cn`

Hao Yu
Fujitsu R&D Center Co., Ltd
Beijing 100016, China
`yu@frdc.fujitsu.com`

Abstract

This paper proposes a hybrid Chinese named entity recognition model based on multiple features. It differentiates from most of the previous approaches mainly as follows. Firstly, the proposed Hybrid Model integrates coarse particle feature (POS Model) with fine particle feature (Word Model), so that it can overcome the disadvantages of each other. Secondly, in order to reduce the searching space and improve the efficiency, we introduce heuristic human knowledge into statistical model, which could increase the performance of NER significantly. Thirdly, we use three sub-models to respectively describe three kinds of transliterated person name, that is, Japanese, Russian and Euramerican person name, which can improve the performance of PN recognition. From the experimental results on People's Daily testing data, we can conclude that our Hybrid Model is better than the models which only use one kind of features. And the experiments on MET-2 testing data also confirm the above conclusion, which show that our algorithm has consistence on different testing data.

1 Introduction

Named Entity Recognition (NER) is one of the key techniques in the fields of Information Extraction, Question Answering, Parsing, Metadata Tagging in Semantic Web, etc. In MET-2 held in conjunction with the Seventh Message Understanding Conference (MUC-7), the task of NER is defined as recognizing seven sub-categories entities: person (PN), location (LN), organization (ON), time, date, currency and percentage. As for Chinese NEs, we further divide PN into five sub-classes, that is, Chinese PN (CPN), Japanese PN (JPN), Russian PN (RPN), Euramerican PN (EPN) and abbreviated PN (APN) like "吴先生/Mr. Wu". Similarly, LN is split into common LN (LN) like "中关村/Zhongguancun" and abbreviated LN (ALN) such as "京/Beijing", "沪/Shanghai". The recognition of time (TM) and numbers (NM) is comparatively simpler and can be implemented via finite state automata. Therefore, our research focuses on the recognition of CPN, JPN, RPN, EPN, APN, LN, ALN and ON.

Compared to English NER, Chinese NER is more difficult. We think that the main differences between Chinese NER and English NER lie in: (1) Unlike English, Chinese lacks the capitalization information which can play very important roles in identifying named entities. (2) There is no space between words in Chinese, so we have to segment the text before NER. Consequently, the errors in word segmentation will affect the result of NER.

In this paper, we proposes a hybrid Chinese NER model based on multiple features which emphasizes on (1) combining fine particle features (Word Model) with coarse particle features (POS Model); (2) integrating human knowledge into statistical model; (3) and using diverse sub-models for different kinds of entities. Especially, we divide transliterated person name into three sub-classes according to their characters set, that is, JPN, RPN and EPN. In order to deduce the complexity of the model and the searching space, we divide the rec-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 427–434, Vancouver, October 2005. ©2005 Association for Computational Linguistics

ognition process into two steps: (1) word segmentation and POS tagging; (2) named entity recognition based on the first step.

Trained on the NEs labeled corpus of five-month People's Daily corpus and tested on one-month People's Daily corpus, the Hybrid Model achieves the following performance. The precision and the recall of PN (including CPN, JPN, RPN, EPN, AP N), LN (including ALN) and ON are respectively (94.06%, 95.21%), (93.98%, 93.48%), and (84.69%, 86.86%). From the experimental results on People's Daily testing data, we can conclude that our Hybrid Model is better than other models which only use one kind of features. And the experiments on MET-2 testing data also confirm the above conclusion, which show that our algorithm has consistence on different testing data.

2 Related Work

On the impelling of international evaluations like MUC, CoNLL, IEER and ACE, the researches on English NER have achieved impressive results. For example, the best English NER system[Chinchor. 1998] in MUC7 achieved 95% precision and 92% recall. However, Chinese NER is far from mature. For example, the performance (precision, recall) of the best Chinese NER system in MET-2 is (66%, 92%), (89%, 91%), (89%, 88%) for PN, LN and ON respectively.

Recently, approaches for NER are a shift away from handcrafted rules[Grishman, et al. 1995] [Krupka, et al. 1998][Black et al. 1998] towards machine learning algorithms, i.e. unsupervised model like DL-CoTrain, CoBoost[Collins, 1999, 2002], supervised learning like Error-driven [Aberdeen, et al. 1995], Decision Tree [Sekine, et al. 1998], HMM[Bikel, et al. 1997] and Maximum Entropy[Borthwick, et al. 1999][Mikheev, et al.1998].

Similarly, the models for Chinese NER can also be divided into two categories: Individual Model and Integrated Model.

Individual Model[Chen, et al. 1998][Sun, et al. 1994][Zheng, et al. 2000] consists of several sub-models, each of them deals with a kind of entities. For example, the recognition of PN may be statistical-based model, while LN and ON may be rule-based model like [Chen, et al. 1998]. Integrated Model[Sun, et al. 2002] [Zhang, et al. 2003][Yu, et al. 1998][Chua, et al. 2002] deals with all kinds of entities in a unified statistical framework. Most of these integrated models can be viewed as a HMM model. The differences among them are the definition of state and the features used in entity model and context model.

In fact, a NER model recognizes named entities through mining the intrinsic features in the entities and the contextual features around the entities. Most of existing approaches employ either coarse particle features, like POS and ROLE[Zhang, et al. 2003], or fine particle features like word. The data sparseness problem is serious if only using fine particle features, and coarse particle features will lose much important information though without serious data sparseness problem. *Our idea is that coarse particle features should be integrated into fine particle features to overcome the disadvantages of them.* However, most systems do not combine them and especially ignore the impact of POS.

Inspired by the algorithms of identifying BaseNP and Chunk[Xun, et al. 2000], we propose a hybrid NER model which emphasizes on combining coarse particle features (POS Model) with fine particle features (Word Model). Though the Hybrid Model can overcome the disadvantages of the Word Model and the POS Model, there are still some problems in such a framework. Data sparseness still exists and very large searching space in decoding will influence efficiency. *Our idea is that heuristic human knowledge can not only improve the time efficiency, but also solve the data sparseness problem to some extent by restricting the generation of entity candidates.* So we intend to incorporate human knowledge into the statistical model to improve efficiency and effectivity of the Hybrid Model.

Similarly, for capturing intrinsic features in different types of entities, we design several sub-models for each kind of entities. For example, we divide transliterated person name into three sub-classes according to their characters sets, that is, JPN, RPN and EPN.

3 Chinese NER with Multiple Features

Chinese NEs have very distinct word features in their composition and contextual information. For example, about 365 highest frequently used surnames cover 99% Chinese surnames[Sun, et al. 1994]. Similarly the characters used for transliterated names are also limited. LNs and ONs often end with the specific words like "省/province" and "公司/company". However, data sparseness is very serious when using word features. So we try to introduce coarse particle feature to overcome the data sparseness problem. POS features are simplest and easy to obtain. Therefore, our hybrid model combines word feature with POS feature to recognize Chinese NEs.

Given a word/pos sequence as equation (1):

$$W/T = w_1/t_1 \cdots w_i/t_i \cdots w_n/t_n \qquad (1)$$

where n is the number of words and t_i is the POS of word w_i. The task of Chinese NE identification is to find the optimal sequence WC^*/TC^* by splitting, combining and classifying the sequence of (1).

$$WC^*/TC^* = wc_1/tc_2 \cdots wc_i/tc_i \cdots wc_m/tc_m \qquad (2)$$

where $wc_i = \lfloor w_j \cdots w_{j+l} \rfloor$, $tc_i = \lfloor t_j \cdots t_{j+l} \rfloor$, $m \le n$.

Note that the definition of words in $\{w_i\}$ set is that each kind of NEs (including PN, APN, LN, ALN, ON, TM, NM) is defined as a word and all the other words in the vocabulary are also defined as individual words. Consequently, $\{w_i\}$ set has $|V|+7$ words, where $|V|$ is the size of vocabulary. The size of $\{t_i\}$ set is 48 which include PKU POS tagging set1 and each kind of NEs.

Obviously, we could obtain the optimal sequence WC^*/TC^* through the following three models: the Word Model, the POS Model and the Hybrid Model.

The Word Model employs word features for NER, which is introduced by [Sun, et al. 2002]. The POS Model employs POS features for NER. This paper proposes a Hybrid Model which combines word features with POS features.

We will describe these models in detail in following section.

3.1 The Hybrid Model

For the convenience of description, we take apart equation (1) into two components: word sequence as equation (3) and POS sequence as (4).

$$W = w_1\, w_2 \cdots w_i \cdots w_n \qquad (3)$$

$$T = t_1\, t_2 \cdots t_i \cdots t_n \qquad (4)$$

The Word Model estimates the probability of generating a NE from the viewpoint of word sequence, which can be expressed in equation (5).

$$WC^* = argmax_{wc} P(WC)P(W \mid WC) \qquad (5)$$

The POS Model estimates the probability of generating a NE from the viewpoint of POS sequence, which can be expressed in equation (6).

$$TC^* = argmax_{TC} P(TC)P(T \mid TC) \qquad (6)$$

Our proposed Hybrid Model combines the Word Model with the POS Model, which can be expressed in the equation (7).

$$\begin{aligned}
&(WC^*, TC^*) \\
&= argmax_{(WC,TC)} P(WC, TC \mid W, T) \\
&= argmax_{(WC,TC)} P(WC, TC, W, T)P(W, T) \\
&= argmax_{(WC,TC)} P(WC, TC, W, T) \\
&\approx argmax_{(WC,TC)} P(W \mid WC)P(WC)[P(T \mid TC)P(TC)]^{\xi}
\end{aligned} \qquad (7)$$

where factor $\zeta > 0$ is to balance the Word Model and the POS Model.

Therefore, the Hybrid Model consists of four sub-models: word context model *P(WC)*, POS context model *P(TC)*, word entity model *P(W|WC)* and POS entity model *P(T|TC)*.

3.2 Context Model

The word context model and the POS context model estimate the probability of generating a word or a POS given previous context. *P(WC)* and *P(TC)* can be estimated according to (8) and (9) respectively.

$$P(WC) = \prod_{i=1}^{m} P(wc_i \mid wc_{i\ 2}\ wc_{i\ 1}) \qquad (8)$$

$$P(TC) = \prod_{i=1}^{m} P(tc_i \mid tc_{i\ 2}\ tc_{i\ 1}) \qquad (9)$$

3.3 Word Entity Model

Different types of NEs have different structures and intrinsic characteristics. Therefore, a single model can't capture all types of entities. Typical, character-based model is more appropriate for PNs, whereas, word-based model is more competent for LNs and ONs. Especially, we divided transliterated PN into three categories such as JPN, RPN and EPN.

For the sake of estimating the probability of generating a NE, we define 19 sub-classes shown as Table 1 according to their position in NEs.

[1] http://icl.pku.edu.cn/nlp-tools/catetkset.html

Tag	Description
Sur	Surname of CPN
Dgb	First character of Given Name of CPN
Dge	Last character of Give Name of CPN
Bfn	First character of EPN
Mfn	Middle character of EPN
Efn	Last character of EPN
RBfn	First character of RPN
RMfn	Middle character of RPN
REfn	Last character of RPN
JBfn	surname of JPN
JMfn	Middle character of JPN
JEfn	Last character of JPN
Bol	First word of LN
Mol	Middle word of LN
Eol	Last word of LN
Aloc	Single character LN
Boo	First word of ON
Moo	Middle word of ON
Eoo	Last word of ON

Table 1 Sub-classes in Entity Model

3.3.1 Word Entity Model for PN

For the class of PN (including CPN, APN, JPN, RPN and EPN), the word entity model is a character-based trigram model which can be expressed in equation (10).

$$\begin{aligned} & P\left(w_{wc_{i1}} \cdots w_{wc_{ik}} \mid wc_i\right) \\ & = P\left(w_{wc_{i1}} \cdots w_{wc_{ik}} \mid BNe \overbrace{MNe \cdots MNe}^{k-2} ENe \right) \\ & \cong P\left(w_{wc_{i1}} \mid BNe\right) \times \prod_{l=2}^{k-1} P\left(w_{wc_{il}} \mid MNe, w_{wc_{i(l-1)}}\right) \\ & \times P\left(w_{wc_{ik}} \mid ENe, w_{wc_{i(k-1)}}\right) \end{aligned} \qquad (10)$$

where, BNe, MNe and ENe denotes the first, middle and last characters respectively.

The word entity models for PN are estimated with Chinese, Japanese, Russian and Euramerican names lists which contain 15.6 million, 0.15 million, 0.44 million, 0.4 million entities respectively.

3.3.2 Word Entity Model for LN and ON

For the class of LN and ON, the word entity model is a word-based trigram model. The model can be expressed by (11).

$$\begin{aligned} & P\left(w_{wc_i\ start} \cdots w_{wc_i\ end} \mid wc_i\right) \\ & = P\left(wc_{wc_{i1}} \cdots wc_{wc_{il}} \cdots wc_{wc_{ik}} \mid BNe \overbrace{MNe \cdots MNe}^{k\ 2} ENe \right) \\ & = P\left(wc_{wc_{i1}} \mid BNe\right) P\left(w_{wc_{i1}\ start} .. w_{wc_{i1}\ end} \mid wc_{wc_{i1}}\right) \\ & \times \prod_{l=2}^{k\ 1} P\left(wc_{il} \mid MNe, wc_{i(l\ 1)}\right) P\left(w_{wc_{il}\ start} \cdots w_{wc_{il}\ end} \mid wc_{wc_{il}}\right) \\ & \times P\left(wc_{wc_{ik}} \mid ENe, wc_{wc_{i(k\ 1)}}\right) P\left(w_{wc_{ik}\ start} \cdots w_{wc_{ik}\ end} \mid wc_{ik}\right) \end{aligned} \qquad (11)$$

The word entity models and the POS entity model for LN and ON are estimated with LN and ON names lists which respectively contain 0.44 mil-lion and 3.2 million entities.

3.3.3 Word Entity Model for ALN

For the class of ALN, we use word-based bi-gram model. The entity model for ALN can be expressed by equation (12).

$$P\left(w_i \mid ALoc\right) = \frac{C\left(w_i, ALoc\right)}{C(ALoc)} \qquad (12)$$

where w_i is the ALN which includes single and multiple characters ALN.

3.4 POS Entity Model

But for the class of PN, it's very difficult to obtain the corpus to train POS Entity Model. For the sake of simplification, we use word entity model shown in equation (10) to replace the POS entity model.

For the class of LN and ON, POS entity model can be expressed by equation (13).

$$\begin{aligned} & P\left(t_{tc_i\ start} \cdots t_{tc_i\ end} \mid tc_i\right) \\ & = P\left(tc_{tc_{i1}} \cdots tc_{tc_{il}} \cdots tc_{tc_{ik}} \mid BNe \overbrace{MNe \cdots MNe}^{k\ 2} ENe \right) \\ & = P\left(tc_{tc_{i1}} \mid BNe\right) P\left(t_{tc_{i1}\ start} .. t_{wc_{i1}\ end} \mid tc_{tc_{i1}}\right) \\ & \times \prod_{l=2}^{k\ 1} P\left(tc_{il} \mid MNe, tc_{i(l\ 1)}\right) P\left(t_{tc_{il}\ start} \cdots t_{tc_{il}\ end} \mid tc_{tc_{il}}\right) \\ & \times P\left(tc_{tc_{ik}} \mid ENe, tc_{tc_{i(k\ 1)}}\right) P\left(t_{tc_{ik}\ start} \cdots t_{tc_{ik}\ end} \mid tc_{ik}\right) \end{aligned} \qquad (13)$$

While for the class of ALN, POS entity model is shown as equation (14).

$$P\left(t_i \mid ALoc\right) = \frac{C\left(ti, ALoc\right)}{C(ALoc)} \qquad (14)$$

4 Heuristic Human Knowledge

In this section, we will introduce heuristic human knowledge that is used for Chinese NER and the method of how to incorporate them into statistical model which are shown as follows.

1. CPN surname list (including 476 items) and JPN surnames list (including 9189 items): Only those characters in the surname list can trigger person name recognition.

2. RPN and EPN characters lists: Only those consecutive characters in the transliterated character list form a candidate transliterated name.

3. Entity Length Restriction: Person name cannot span any punctuation and the length of CN cannot exceed 8 characters while the length of TN is unrestrained.

4. Location keyword list (including 607 items): If the word belongs to the list, 2~6 words before the salient word are accepted as candidate LNs.

5. General word list (such as verbs and prepositions): Words in the list usually is followed by a location name, such as "在/at", "去/go". If the current word is in the list, 2~6 words following it are accepted as candidate LNs.

6. ALN name list (including 407 items): If the current word belongs to the list, we accept it as a candidate ALN.

7. Organization keyword list (including 3129 items): If the current word is in organization keyword list, 2~6 words before keywords are accepted as the candidate ONs.

8. An organization name template list: We mainly use organization name templates to recognize the missed nested ONs in the statistical model. Some of these templates are as follows:

ON-->LN D OrgKeyWord*
ON-->PN D OrgKeyWord*
ON-->ON OrgKeyWord

D and *OrgKeyWord* denote words in the middle of ONs and ONs keywords. *D** means repeating zero or more times.

5 Back-off Model to Smooth

Data sparseness problem still exists. As some parameters were never observed in training corpus, the model will back off to a less powerful model. The escape probability[Black, et al. 1998] was adopted to smooth the statistical model shown as (15).

$$\hat{p}(W_N|W_1\cdots W_{N\ 1}) = \lambda_N p(W_N|W_1\cdots W_{N\ 1}) + \lambda_{N\ 1}p(W_N|W_2\cdots W_{N\ 1}) + \cdots + \lambda_1 p(W_N) + \lambda_0 p_0 \quad (15)$$

where $\lambda_N = 1\quad e_N$, $\lambda_i = (1\quad e_i)\sum_{k=i+1}^{N} e_k, 0 < i < N$, and e_i is the escape probability which can be estimated by equation (16).

$$e_N = \frac{q(W_1W_2\cdots W_{N\ 1})}{f(W_1W_2\cdots W_{N\ 1})} \quad (16)$$

$q(w_1w_2...w_{N-1})$ in (16) denotes the number of different symbol w_N that have directly followed the word sequence $w_1w_2...w_{N-1}$.

6 Experiments

In this chapter, we will conduct experiments to answer the following questions.

Will the Hybrid Model be more effective than the Word Model and the POS Model? To answer this question, we will compare the performances of models with different parameter ζ and find the best value of ζ in equation (7).

Will the conclusion from different testing sets be consistent? To answer this question, we evaluate models on the MET-2 test data and compare the performances of the Word Model, the POS Model and the Hybrid Model.

Will the performance be improved significantly after combining human knowledge? To answer this question, we compare two models with and without human knowledge.

In our evaluation, only NEs with correct boundaries and correct categories are considered as the correct recognition. We conduct evaluations in terms of precision, recall and F-Measure. Note that PNs in experiments includes all kinds of PNs and LNs include ALNs.

6.1 Will the Hybrid Model be More Effective Than the Word Model and POS Model?

The parameter ζ in equation (7) denotes the balancing factor of the Word Model and the POS Model. The larger ζ, the larger contribution of the POS Model. The smaller ζ, the larger contribution of the Word Model. So the task of this experiment is to find the best value of ζ. In this experiment, the training corpus is from five-month's People's Daily tagged with NER tags and the testing set is from one-month's People's Daily.

With the change of ζ, the performances of recognizing PNs are shown in Fig.1.

Note that the left, middle and right point in abscissa respectively denote the performance of the

Word Model, the Hybrid Model and the POS Model.

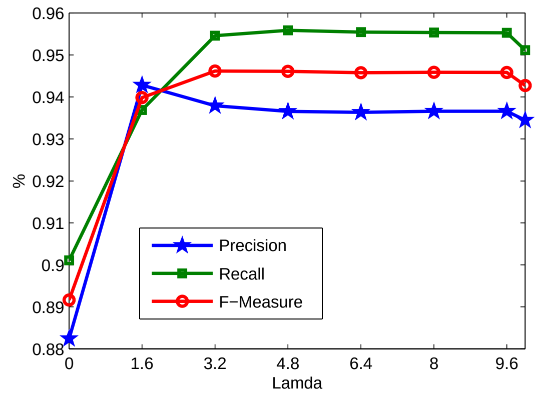


Fig.1 Performance of Recognizing LNs Impacted by ζ

From Fig.1, we can find that the performances of recognizing PNs are improved with the increasing of ζ in the beginning stage but decline in the ending. This experiment shows that the Word Model and the POS Model can overcome their disadvantages, and it is a feasible approach to integrate the Word Model and the POS Model in order to improve the performance PNs recognition.

With the change of ζ, the performances of recognizing LNs are shown in Fig.2.

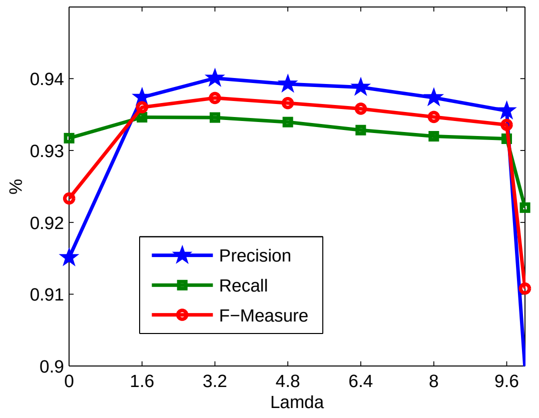


Fig.2 Performance of Recognizing LNs Impacted by ζ

As the Fig.2 shows, the precision and recall of LNs are improved with the increasing of ζ and decreased in the later stage. This phenomenon also proves that the Hybrid Model is better for recognizing LN than either the Word Model or the POS Model.

Similarly, with the change of ζ, the performances of recognizing ONs are shown in Fig.3.

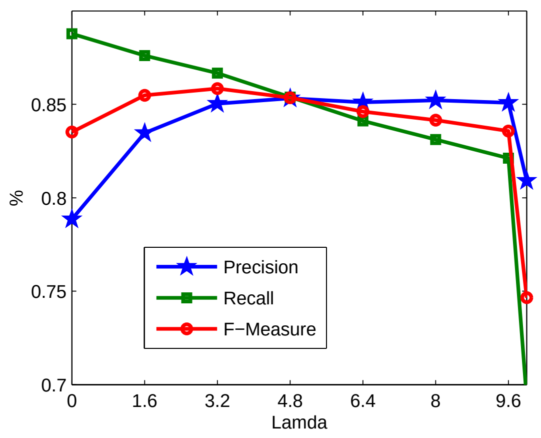


Fig.3 Performance of Recognizing LNs Impacted by ζ

Comparing Fig.3 with Fig.1 and Fig.2, we find that the POS Model has different impact on recognizing ONs from that on recognizing PNs and LNs. Especially, the POS Model has obvious side-effect on the recall. We speculate that the reasons may be that the probability of generating POS sequence by POS entity model is lower than that by POS context model.

According to Fig.1~Fig.3, we choose the best value ζ = 2.8. And the performances of different models are shown in Table 2 in detail.

		P(%)	R(%)	F(%)
Hybrid Model (ζ= 2.8)	**PN**	94.06	95.21	94.63
	LN	93.98	93.48	93.73
	ON	84.69	86.86	85.76
Word Model				
	PN	88.24	90.11	89.16
	LN	91.50	93.17	92.32
	ON	78.85	88.77	83.52
POS Model				
	PN	93.44	95.11	94.27
	LN	89.97	92.20	91.07
	ON	80.90	69.29	74.65

Table 2 Performance of the Hybrid Model, the Word Model and the POS Model

From Table 2, we find that the F-Measures of the Hybrid Model for PN, LN, ON are improved by 5.4%, 1.4%, 2.2% respectively in comparison with the Word Model, and these F-Measures are improved by 0.4%, 2.7%, 11.1% respectively in comparison with the POS Model.

Conclusion 1: The experimental results validate our idea that the Hybrid Model can improve the performance of both the Word Model and the POS Model. However, the improvements for PN, LN and ON are different. That is, the POS Model has obvious side-effect on the recall of ON recognition at all times, while the recalls for PN and ON recognition are improved in the beginning but decreased in the ending with the increasing of ζ.

6.2 Will the Conclusion from Different Testing Sets be Consistent?

We also conduct experiments on the MET-2 testing corpus to validate our conclusion from Exp.1, that is, the Hybrid Model could achieve better performance than either the Word Model or the POS Model alone. The experimental results (F-Measure) on MET-2 are shown in Table 3.

Model	**Word Model**	**Hybrid Model**	**POS Model**
PN	75.21%	**80.77**%	76.61%
LN	89.78%	**90.95**%	89.81%
ON	76.30%	**80.21**%	76.83%

Table 3 F-Measure on MET-2 test corpus

Comparing Table 3 with Table 2, we find that the performances of models on MET-2 are not as good as that on People Daily's testing data. The main reason lies in that the NE definitions in People Daily's corpus are different from that in MET-2. However, Table 3 can still validate our conclude 1, that is, the Hybrid Model is better than both the Word Model and the POS Model. For example, the F-Measures of the Hybrid Model for PN, LN and ON are improved by 5.6%, 1.2% and 3.9% respectively in comparison with the Word Model, and these F-Measures are improved by 4.2%, 3.1% and 3.4% respectively in comparison with the POS Model.

Conclusion 2: Though the performances of the Hybrid Model on MET-2 are not as good as that on People's Daily corpus, the experimental results also support conclusion 1, i.e. the Hybrid Model which combining the Word Model with the POS Model can achieve better performance than either the Word Model or the POS Model.

6.3 Will the Performance be Improved Significantly after Incorporating Human Knowledge?

One of our ideas in this paper is that human knowledge can not only reduce the search space, but also improve the performance through avoiding generating the noise NEs. This experiment will be conducted to validate this idea. Table 4 shows the performances of models with and without human knowledge.

		P(%)	R(%)	F(%)
Model I	**PN**	91.81	70.65	79.85
	LN	79.47	88.83	83.89
	ON	64.95	80.63	71.95
Model II				
	PN	94.06	95.21	94.63
	LN	93.98	93.48	93.73
	ON	84.69	86.86	85.76

Table 4 Performances Impacted by Human Knowledge

From Table 4, we find that F-Measure of model with human knowledge (Model II) is improved by 14.8%，9.8%，13.8% for PN, LN and ON respectively compared with that of the model without human knowledge (Model I).

Conclusion 3: From this experiment, we learn that human knowledge can not only reduce the search space, but also significantly improve the performance of pure statistical model.

7 Conclusion

In this paper, we propose a hybrid Chinese NER model which combines multiple features. The main contributions are as follows: ① The proposed Hybrid Model emphasizes on integrating coarse particle feature (POS Model) with fine particle feature (Word Model), so that it can overcome the disadvantages of each other; ② In order to reduce the search space and improve the efficiency of model, we incorporate heuristic human knowledge into statistical model, which could increase the performance of NER significantly; ③ For capturing intrinsic features in different types of entities, we design several sub-models for different entities. Especially, we divide transliterated person name into three sub-classes according to their characters set, that is, CPN JPN, RPN and EPN.

There is a lack of effective recognition strategy for abbreviated ONs such as 昆明机床(Kunming Machine Tool Co.,Ltd), 凤凰光学(Phoenix Photonics Ltd) in this paper. And most of misrecognized ONs in current system belong to them. So in the future work, we will be focusing more on recognizing abbreviated ONs.

8 Acknowledgements

This research is carried out as part of the cooperative project with Fujitsu R&D Center Co., Ltd. We would like to thank Yingju Xia, Fumihito Nisino for helpful feedback in the process of developing and implementing. This work was supported by the Natural Sciences Foundation of China under grant No. 60372016 and 60272041, the Natural Science Foundation of Beijing under grant No. 4052027.

References

N.A. Chinchor: Overview of MUC-7/MET-2. In: Proceedings of the Seventh Message Understanding Conference (MUC-7), April. (1998).

Youzheng Wu, Jun Zhao, Bo Xu: Chinese Named Entity Recognition Combining Statistical Model with Human Knowledge. In: The Workshop attached with 41st ACL for Multilingual and Mix-language Named Entity Recognition, Sappora, Japan. (2003) 65-72.

Endong Xun, Changning Huang, Ming Zhou: A Unified Statistical Model for the Identification of English BaseNP. In: Proceedings of ACL-2000, Hong Kong. (2000).

Jian Sun, Jianfeng Gao, Lei Zhang, Ming Zhou, Changning Huang: Chinese Named Entity Identification Using Class-based Language Model. In: COLING 2002. Taipei, August 24-25. (2002).

Huaping Zhang, Qun Liu, Hongkui Yu, Xueqi Cheng, Shuo Bai: Chinese Named Entity Recognition Using Role Model. In: the International Journal of Computational Linguistics and Chinese Language Processing, vol.8, No.2. (2003) 29-60.

D.M. Bikel, Scott Miller, Richard Schwartz, Ralph Weischedel: Nymble: a High-Performance Learning Name-finder. In: Fifth Conference on Applied Natural Language Processing, (published by ACL). (1997) 194-201.

Borthwick .A: A Maximum Entropy Approach to Named Entity Recognition. PhD Dissertation. (1999).

Mikheev A., Grover C. and Moens M: Description of the LTG System Used for MUC-7. In: Proceedings of 7th Message Understanding Conference (MUC-7), 1998.

Sekine S., Grishman R. and Shinou H: A decision tree method for finding and classifying names in Japanese texts. In: Proceedings of the Sixth Workshop on Very Large Corpora, Canada, 1998.

Aberdeen, John, et al: MITRE: Description of the ALEMBIC System Used for MUC-6. In: Proceedings of the Sixth Message Understanding Conference (MUC-6), November. (1995) 141-155.

Ralph Grishman and Beth Sundheim: Design of the MUC-6 evaluation. In: 6th Message Understanding Conference, Columbia, MD. (1995)

Krupka, G. R. and Hausman, K. IsoQuest: Inc.: Description of the NetOwl TM Extractor System as Used for MUC-7. In Proceedings of the MUC-7, 1998.

Black, W.J.; Rinaldi, F, Mowart, D: FACILE: Description of the NE System Used for MUC-7. In Proceedings of the MUC-7, 1998.

Michael Collins, Yoram Singer: Unsupervised models for named entity classification. In Proceedings of EMNLP. (1999)

Michael Collins: Ranking Algorithms for Named Entity Extraction: Boosting and the Voted Perceptron. In: Proceeding of ACL-2002. (2002) 489-496.

S.Y.Yu, et al: Description of the Kent Ridge Digital Labs System Used for MUC-7. In: Proceedings of the Seventh Message Understanding Conference, 1998.

H.H. Chen, et al: Description of the NTU System Used for MET2. In: Proceedings of the Seventh Message Understanding Conference.

Tat-Seng Chua, et al: Learning Pattern Rules for Chinese Named Entity Extraction. In: Proceedings of AAAI'02. (2002)

Maosong Sun, et al: Identifying Chinese Names in Unrestricted Texts. Journal of Chinese Information Processing. (1994).

Jiahen Zheng, Xin Li, Hongye Tan: The Research of Chinese Names Recognition Methods Based on Corpus. In: Journal of Chinese Information Processing. Vol.14 No.1. (2000).

CoNLL. http://cnts.uia.ac.be/conll2004/

IEER. http://www.nist.gov/speech/tests/ie-er/er99/er99.htm

ACE. http://www.itl.nist.gov/iad/894.01/tests/ace/

Cluster-specific Named Entity Transliteration

Fei Huang

School of Computer Science
Carnegie Mellon University, Pittsburgh, PA 15213
fhuang@cs.cmu.edu

Abstract

Existing named entity (NE) transliteration approaches often exploit a general model to transliterate NEs, regardless of their origins. As a result, both a Chinese name and a French name (assuming it is already translated into Chinese) will be translated into English using the same model, which often leads to unsatisfactory performance. In this paper we propose a cluster-specific NE transliteration framework. We group name origins into a smaller number of clusters, then train transliteration and language models for each cluster under a statistical machine translation framework. Given a source NE, we first select appropriate models by classifying it into the most likely cluster, then we transliterate this NE with the corresponding models. We also propose a phrase-based name transliteration model, which effectively combines context information for transliteration. Our experiments showed substantial improvement on the transliteration accuracy over a state-of-the-art baseline system, significantly reducing the transliteration character error rate from 50.29% to 12.84%.

1 Introduction

Named Entity (NE) translation and transliteration are very important to many multilingual natural language processing tasks, such as machine translation, crosslingual information retrieval and question answering. Although some frequently occurring NEs can be reliably translated using information from existing bilingual dictionaries and parallel or monolingual corpora (Al-Onaizan and Knight, 2002; Huang and Vogel, 2002; Lee and Chang, 2003), less frequently occurring NEs, especially new names, still rely on machine transliteration to generate their translations.

NE machine transliteration generates a phonetically similar equivalent in the target language for a source NE, and transliteration patterns highly depend on the name's origin, e.g., the country or the language family this name is from. For example, when transliterating names[1] from Chinese into English, as shown in the following example, the same Chinese character "金" is transliterated into different English letters according to the origin of each person.

金人庆 --- **Jin** Renqing (China)
金大中 --- **Kim** Dae-jung (Korea)
马丁 路德 金 --- Martin Luther **King** (USA)
金丸信 --- **Kane**maru Shin (Japan)
何塞 华金 布伦纳 --- Jose Joa**quin** Brunner (Chile)

Several approaches have been proposed for name transliteration. (Knight and Graehl, 1997) proposed a generative transliteration model to transliterate foreign names in Japanese back to English using finite state transducers. (Stalls and Knight, 1998) expanded that model to Arabic-English transliteration. (Meng et al. 2001) developed an English-Chinese NE transliteration technique using pronunciation lexicon and phonetic mapping rules. (Virga and Khudanpur, 2003) applied statistical machine translation models to "translate" English names into Chinese characters for Mandarin spoken document retrieval. All these approaches exploit a general model for NE transliteration, where source names from different origins or language families are transliterated into the target language with the same rules or probability distributions, which fails to capture their different

[1] Assuming foreign names are already transliterated into Chinese.

transliteration patterns. Alternatively, (Qu and Grefenstette, 2004) applied language identification of name origins to select language-specific transliterations when back-transliterating Japanese names from English to Japanese. However, they only classified names into three origins: Chinese, Japanese and English, and they used the Unihan database to obtain the mapping between kenji characters and romanji representations.

Ideally, to explicitly model these transliteration differences we should construct a transliteration model and a language model for each origin. However, some origins lack enough name translation pairs for reliable model training. In this paper we propose a cluster-specific NE transliteration framework. Considering that several origins from the same language family may share similar transliteration patterns, we group these origins into one cluster, and build cluster-specific transliteration and language models.

Starting from a list of bilingual NE translation pairs with labeled origins, we group closely related origins into clusters according to their language and transliteration model perplexities. We train cluster-specific language and transliteration models with merged name translation pairs. Given a source name, we first select appropriate models by classifying it into the most likely cluster, then we transliterate the source name with the corresponding models under the statistical machine translation framework. This cluster-specific transliteration framework greatly improves the transliteration performance over a general transliteration model. Further more, we propose a phrase-based transliteration model, which effectively combines context information for name transliteration and achieves significant improvements over the traditional character-based transliteration model.

The rest of the paper is organized as following: in section 2 we introduce the NE clustering and classification schemes, and we discuss the phrase-based NE transliteration in section 3. Experiment settings and results are given in section 4, which is followed by our conclusion.

2 Name Clustering and Classification

Provided with a list of bilingual name translation pairs whose origins are already labeled, we want to find the origin clusters where closely related origins (countries sharing similar languages or cultural heritages) are grouped together.

We define the similarity measure between two clusters as their LM and TM perplexities. Let $S_i = \{(F_i, E_i)\}$ denote a set of name translation pairs from origin i, from which model θ_i is trained: $\theta_i = (P_{c(i)}, P_{e(i)}, P_{t(i)})$. Here $P_{c(i)}$ and $P_{e(i)}$ are N-gram character language models (LM) for source and target languages, and $P_{t(i)}$ is a character translation model trained based on IBM translation model 1 (Brown et.al. 1993). The distance between origin i and origin j can be symmetrically defined as:

$$d(i,j) = -\frac{1}{|S_i|}\log P(S_i \mid \theta_j) - \frac{1}{|S_j|}\log P(S_j \mid \theta_i),$$

where, assuming name pairs are generated independently,

$$P(S_i \mid \theta_j) \propto \sum_{t=1}^{|S_i|} \log[P_{c(j)}(F_i^t)P_{t(j)}(E_i^t \mid F_i^t) + P_{e(j)}(E_i^t)P_{t(j)}(F_i^t \mid E_i^t)]$$

We calculate the pair-wise distances among these origins, and cluster them with group-average agglomerative clustering. The distance between clusters C_i and C_j is defined as the average distance between all origin pairs in each cluster. This clustering algorithm initially sets each origin as a single cluster, then recursively merges the closest cluster pair into one cluster until an optimal number of clusters is formed.

Among all possible cluster configurations, we select the optimal cluster number based on the model perplexity. Given a held-out data set L, a list of name translation pairs from different origins, the probability of generating L from a cluster configuration Θ_ω is the product of generating each name pair from its most likely origin cluster:

$$P(L \mid \Theta_\omega) = \prod_{t=1}^{|L|} \max_{j \in \Theta_\omega} P(F^t, E^t \mid \theta_j)P(\theta_j)$$
$$= \prod_{t=1}^{|L|} \max_{j \in \Theta_\omega} P_{c(j)}(F^t)P_{e(j)}(E^t)P(\theta_j)$$

We calculate the language model perplexity:

$$pp(L, \Theta_\omega) = 2^{-\frac{1}{|L|}\log P(L \mid \Theta_\omega)} = P(L \mid \Theta_\omega)^{-1/|L|},$$

and select the model configuration with the smallest perplexity. We clustered 56K Chinese-English name translation pairs from 112 origins, and evaluate the perplexities of different models (number of

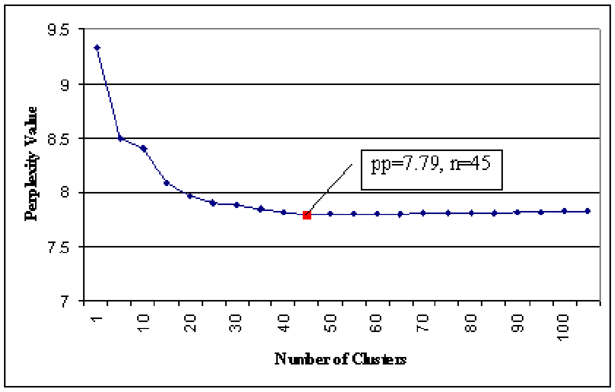


Figure 1. Perplexity value of LMs with different number of clusters

clusters) with regard to a held-out 3K name pairs. As shown in Figure 1, the perplexity curve reaches its minimum when $n = 45$. This indicates that the optimal cluster number is 45.

Table 1 lists some typical origin clusters. One may notice that countries speaking languages from the same family are often grouped together. These countries are either geographically adjacent or historically affiliated. For example, in the English cluster, the Netherlands (Dutch) seems an abnormality. In the clustering process it was first grouped with the South Africa, which was colonized by the Dutch and the English in the seventeenth century. This cluster was further grouped into the English-speaking cluster. Finally, some origins cannot be merged with any other clusters because they have very unique names and translation patterns, such as China and Japan, thus they are kept as single origin clusters.

For name transliteration task, given a source name *F* we want to classify it into the most likely cluster, so that the appropriate cluster-specific model can be selected for transliteration. Not knowing *F*'s translation *E*, we cannot apply the translation model and the target language model for name origin classification. Instead we train a Bayesian classifier based on N-gram source character language models, and assign the name to the cluster with the highest LM probability. Assuming a source name is composed of a sequence of source characters: $F = \{f_1, f_2, ..., f_l\}$. We want to find the cluster j^* such that

Arabic	Afghanistan, Algeria, Egypt, Iran, Iraq, Jordan, Kuwait, Pakistan, Palestine, Saudi Arabia, Sudan, Syria, Tunisia, Yemen, …
Spanish-Portuguese	Angola, Argentina, Bolivia, Brazil, Chile, Colombia, Cuba, Ecuador, Mexico, Peru, Portugal, Spain, Venezuela, …
English	Australia, Canada, Netherlands, New Zealand, South Africa, UK, USA, …
Russian	Belarus, Kazakhstan, Russia, Ukraine
East European	Bosnia and Herzegovina, Croatia, Yugoslavia
French (African)	Benin, Burkina Faso, Cameroon, Central African Republic, Congo, Gabon, Ivory Coast
German	Austria, Germany, Switzerland
French	Belgium, France, Haiti
Korean	North Korea, South Korea
Danish-Swedish	Denmark, Norway, Sweden
Single Clusters	China Japan Indonesia Israel ……

Table 1 Typical name clusters (n=45)

$$\begin{aligned} j^* &= \arg\max_j P(\theta_j \mid F) \\ &= \arg\max_j P(\theta_j) P(F \mid \theta_j) \qquad (1) \\ &= \arg\max_j P(\theta_j) P_{c(j)}(F) \end{aligned}$$

where $P(\theta_j)$ is the prior probability of cluster *j*, estimated based on its distribution in all the training data, and $P_{c(j)}(F)$ is the probability of generating this source name based on cluster j's character language model.

3 Phrase-Based Name Transliteration

Statistical NE transliteration is similar to the statistical machine translation in that an NE translation pair can be considered as a parallel sentence pair, where "words" are characters in source and target languages. Due to the nature of name transliteration, decoding is mostly monotone.

NE transliteration process can be formalized as:

$$E^* = \text{argmax}_E\, P(E \mid F) = \text{argmax}_E\, P(F \mid E)P(E)$$

where E^* is the most likely transliteration for the source NE F, $P(F|E)$ is the transliteration model and $P(E)$ is the character-based target language model. We train a transliteration model and a language model for each cluster, using the name translation pairs from that cluster.

3.1 Transliteration Model

A transliteration model provides a conditional probability distribution of target candidates for a given source transliteration unit: a single character or a character sequence, i.e., "phrase". Given enough name translation pairs as training data, we can select appropriate source transliteration units, identify their target candidates from a character alignment path within each name pair, and estimate their transliteration probabilities based on their co-occurrence frequency.

A naive choice of source transliteration unit is a single character. However, single characters lack contextual information, and their combinations may generate too many unlikely candidates. Motivated by the success of phrase-based machine translation approaches (Wu 1997, Och 1999, Marcu and Wong 2002 and Vogel et. al., 2003), we select transliteration units which are long enough to capture contextual information while flexible enough to compose new names with other units. We discover such source transliteration phrases based on a character collocation likelihood ratio test (Manning and Schutze 1999). This test accepts or rejects a null hypothesis that the occurrence of one character f_1 is independent of the other, f_2, by calculating the likelihood ratio between the independent (H_0) and dependent (H_1) hypotheses:

$$\begin{aligned}\log\lambda &= \log\frac{L(H_0)}{L(H_1)}\\ &= \log L(c_{12}, c_1, p) + \log L(c_2 - c_{12}, N - c_1, p)\\ &\quad - \log L(c_{12}, c_1, p_1) - \log L(c_2 - c_{12}, N - c_1, p_2)\end{aligned}$$

L is the likelihood of getting the observed character counts under each hypothesis. Assuming the character occurrence frequency follows a binomial distribution, $L(k,n,x) = \binom{n}{k} x^k (1-x)^{n-k}$,

c_1, c_2, c_{12} are the frequencies of f_1, f_2 and $f_1 \wedge f_2$, and N is the total number of characters. p, p_1 and p_2 are defined as:

$$p = \frac{c_2}{N}, \quad p_1 = \frac{c_{12}}{c_2}, \quad p_2 = \frac{c_2 - c_{12}}{N - c_1}.$$

We calculate the likelihood ratio for any adjacent source character pairs, and select those pairs whose ratios are higher than a predefined threshold. Adjacent character bigrams with one character overlap can be recursively concatenated to form longer source transliteration phrases. All these phrases and single characters are combined to construct a cluster-specific phrase segmentation vocabulary list, T. For each name pair in that cluster, we

1. Segment the Chinese character sequence into a source transliteration phrase sequence based on maximum string matching using T;
2. Convert Chinese characters into their romanization form, *pinyin*, then align the pinyin with English letters via phonetic string matching, as described in (Huang et. al., 2003);
3. Identify the initial phrase alignment path based on the character alignment path;
4. Apply a beam search around the initial phrase alignment path, searching for the optimal alignment which minimizes the overall phrase alignment cost, defined as:

$$A^* = \arg\min_A \sum_{a_i \in A} D(f_i, e_{a_i}).$$

Here f_i is the ith source phrase in F, e_{a_i} is its target candidate under alignment A. Their alignment cost D is defined as the linear interpolation of the phonetic transliteration cost $\log P_{trl}$ and semantic translation cost $\log P_{trans}$:

$$D(f,e) = \lambda \log P_{trl}(e \mid f) + (1-\lambda)\log P_{trans}(e \mid f),$$

where P_{trl} is the product of the letter transliteration probabilities over aligned pinyin-English letter pairs, P_{trans} is the phrase translation probability calculated from word translation probabilities, where a "word" refers to a Chinese character or a English letter. More details about these costs are described in (Huang et. al., 2003). λ is a cluster-specific interpolation weight, reflecting the relative contributions of the transliteration cost and the translation cost. For example, most Latin language names are often phonetically translated into Chinese, thus the transliteration cost is usually the dominant feature. However, Japanese names are often semantically translated when they contain characters borrowed from Chinese, therefore the translation cost is more important for the Japanese model (λ=0 in this case). We empirically select the interpolation weight for each cluster, based on their transliteration performance on held-out name pairs, and the combined model with optimal interpolation weights achieves the best overall performance.

We estimate the phrase transliteration probability according to their normalized alignment frequencies. We also include frequent sub-name translations (first, middle and last names) in the transliteration dictionary. Table 2 shows some typical transliteration units (characters or phrases) from three clusters. They are mostly names or sub-names capturing cluster-specific transliteration patterns. It also illustrates that in different clusters the same character has different transliteration candidates with different probabilities, which justifies the cluster-specific transliteration modeling.

Arabic	穆罕默德 mohamed 阿卜杜勒 abdul 艾哈迈德 ahmed
	尤: yo (0.27) y(0.19) you(0.14)…
English	约翰 john 威廉 william 彼得 peter
	尤: u(0.25) you(0.38) joo(0.16)…
Russian	弗拉基米尔 vladimir 伊万诺夫 ivanov −耶维奇 -yevich
	尤： yu(0.49) y(0.08) iu(0.07)…

Table 2. Transliteration units examples from three name clusters.

3.2 Language model and decoding

For each cluster we train a target character language model from target NEs. We use the N-gram models with standard smoothing techniques.

During monotone decoding, a source NE is segmented into a sequence of transliteration units, and each source unit is associated with a set of target candidate translations with corresponding probabilities. A transliteration lattice is constructed to generate all transliteration hypotheses, among which the one with the minimum transliteration and language model costs is selected as the final hypothesis.

4 Experiment Results

We selected 62K Chinese-English person name translation pairs for experiments. These origin-labeled NE translation pairs are from the name entity translation lists provided by the LDC[2] (including the who'swho (china) and who'swho (international) lists), and devided into three parts: system training (90%), development (5%) and testing (5%). In the development and test data, names from each cluster followed the same distribution as in the training data.

4.1 NE Classification Evaluation

We evaluated the source name classification accuracy, because classification errors will lead to incorrect model selection, and result in bad transliteration performance in the next step. We trained 45 cluster-specific N-gram source character language models, and classified each source name into the most likely cluster according to formula 1. We evaluated the classification accuracy on a held-out test set with 3K NE pairs. We also experimented with different *N* values. Table 3 shows the classification accuracy, where the 3-gram model achieves the highest classification accuracy. A detailed analysis indicates that some classification errors are due to the inherent uncertainty of some names, e. g, "骆家辉 (Gary Locke)", a Chinese American, was classified as a Chinese name based on its source characters while his origin was labeled as USA.

N=2	N=3	N=4	N=5	N=6	N=7
83.62	**84.88**	84.00	84.04	83.94	83.94

Table 3. Source name origin classification accuracies

[2] http://www.ldc.upenn.edu

4.2 NE Transliteration Evaluation

We first evaluated transliteration results for each cluster, then evaluated the overall results on the whole test set, where a name was transliterated using the cluster-specific model in which it was classified. The evaluation metrics are:

- Top1 accuracy (**Top1**), the percentage that the top1 hypothesis is correct, i.e., the same as the reference translation;
- Top 5 accuracy (**Top5**), the percentage that the reference translation appears in the generated top 5 hypotheses;
- Character error rate (**CER**), the percentage of incorrect characters (inserted, deleted and substituted English letters) when the top 1 hypothesis is aligned to the reference translation.

Our baseline system was a character-based general transliteration model, where 56K NE pairs from all clusters were merged to train a general transliteration model and a language model (**CharGen**). We compare it with a character-based cluster-specific model (**CharCls**) and a phrase-based cluster-specific model (**PhraCls**). The CERs of several typical clusters are shown in Table 4.

Because more than half of the training name pairs are from Latin language clusters, the general transliteration and language models adopted the Latin name transliteration patterns. As a result, it obtained reasonable performance (20-30% CERs) on Latin language names, such as Spanish, English and French names, but strikingly high (over 70%) CERs on oriental language names such as Chinese and Japanese names, even though the Chinese cluster has the most training data.

When applying the character-based cluster-specific models, transliteration CERs consistently decreased for all clusters (ranging from 6.13% relative reduction for the English cluster to 97% for the Chinese cluster). As expected, the oriental language names obtained the most significant error reduction because the cluster-specific models were able to represent their unique transliteration patterns. When we applied the phrased-based transliteration models, CERs were further reduced by 23% ~ 51% for most clusters, because the context information were encapsulated in the transliteration phrases. An exception was the Chinese cluster, where names were often translated according to the pinyin of single characters, thus phrase-based transliteration slightly decreased the performance.

The transliteration performance of different clusters varied a lot. The Chinese cluster achieved 96.09% top 1 accuracy and 1.69% CER with the character-based model, and other clusters had CERs ranging from 7% to 30%. This was partly because of the lack of training data (e.g, for the Japanese cluster), and partly because of unique transliteration patterns of different languages. We try to measure this difference using the average number of translations per source phrase (**AvgTrans**), as shown in Table 4. This feature reflected the transliteration pattern regularity, and seemed linearly correlated with the CERs. For example, compared with the English cluster, Russian names have more regular translation patterns, and its CER is only 1/3 of the English cluster, even with only half size of training data.

In Table 5 we compared translation examples from the baseline system (**CharGen**), the phrase-based cluster-specific system (**PhraCls**) and a online machine translation system, the **BabelFish**[3]. The **CharGen** system transliterated every name in the Latin romanization way, regardless of each name's original language. The **BabelFish** system inappropriately translated source characters based on their semantic meanings, and the results were difficult to understand. The **PhraCls** model captured cluster-specific contextual information, and achieved the best results.

We evaluated three models' performances on all the test data, and showed the result in Table 6. The **CharGen** model performed rather poorly transliterating oriental names, and the overall CER was around 50%. This result was comparable to other state-of-the-art statistical name transliteration systems (Virga and Khudanpur, 2003). The **CharCls** model significantly improved the top1 and top 5 transliteration accuracies from 3.78% to 51.08%, and from 5.84% to 56.50%, respectively. Consistently, the CER was also reduced from 50.29% to 14.00%. Phrase-based transliteration further increased the top 1 accuracy by 9.3%, top 5 accuracy by 10.7%, and reduced the CER by 8%, relatively. All these improvements were statistically significant.

[3] http://babelfish.altavista.com/

Cluster	**Training data size**	**CharGen (CER)**	**CharCls (CER)**	**PhraCls (CER)**	**AvgTrans**
Arabic	8336	22.88	18.93	14.47	4.58
Chinese	27093	76.45	1.69	1.71	3.43
English	8778	31.12	29.21	17.27	5.02
French	2328	27.66	18.81	9.07	3.51
Japanese	2161	86.94	38.65	29.60	7.57
Russian	4407	29.17	9.62	6.55	3.64
Spanish	8267	18.87	15.99	10.33	3.61

Table 4. Cluster-specific transliteration comparison

Cluster	**Source**	**Reference**	**CharGen**	**PhraCls**	**BabelFish**
Arabic	纳吉 萨布里 艾哈迈德	Nagui Sabri Ahmed	Naji Saburi Ahamed	Naji Sabri Ahmed	In natrium 吉 萨 cloth Aihamaide
Chinese	范志伦	Fan Zhilun	Van Tylen	Fan zhilun	Fan Zhilun
English	罗伯特 斯特德沃德	Robert Steadward	Robert Stdwad	Robert Sterdeward	Robert Stead Warder
French	让-吕克 科雷捷	Jean-luc Cretier	Jean-luk Crete	Jean-luc Cretier	Let - Lu Keke lei Jie
Japanese	小林隆治	Kobayashi Ryoji	Felinonge	Kobayashi Takaji	Xiaolin pros-perous gov-erns
Russian	弗拉基米尔 萨姆索诺夫	Vladimir Samsonov	Frakimir Samsonof	Vladimir Samsonov	弗拉基 mil sum rope Knoff
Spanish	鲁道夫 卡多索	Rodolfo Cardoso	Rudouf Cardoso	Rodolfo Cadozo	Rudolph card multi- ropes

Table 5. Transliteration examples from some typical clusters

Model	Top1 (%)	Top5 (%)	CER (%)
CharGen	3.78±0.69	5.84±0.88	50.29±1.21
CharCls	51.08±0.84	56.50±0.87	14.00±0.34
PhraCls	56.00±0.84	62.66±0.91	12.84±0.41

Table 6 Transliteration result comparison

5 Conclusion

We have proposed a cluster-specific NE transliteration framework. This framework effectively modeled the transliteration differences of source names from different origins, and has demonstrated substantial improvement over the baseline general model. Additionally, phrase-based transliteration further improved the transliteration performance by a significant margin.

References

Y. Al-Onaizan and K. Knight. 2002. Translating named entities using monolingual and bilingual resources. *In Proceedings of the ACL-2002,* pp400-408, Philadelphia, PA, July, 2002.

F. Huang and S. Vogel. 2002. Improved Named Entity Translation and Bilingual Named Entity Extraction, *Proceedings of the ICMI-2002*. Pittsburgh, PA, October 2002

F. Huang, S. Vogel and A. Waibel. 2003. Automatic Extraction of Named Entity Translingual Equivalence Based on Multi-feature Cost Minimization. *Proceedings of the ACL-2003, Workshop on Multilingual and Mixed Language Named Entity Recognition*. Sapporo, Japan.

K. Knight and J. Graehl. 1997. Machine Transliteration. *Proceedings of the ACL-1997*. pp.128-135, Somerset, New Jersey.

C. J. Lee and J. S. Chang. 2003. Acquisition of English-Chinese Transliterated Word Pairs from Parallel-Aligned Texts using a Statistical Machine Transliteration Model. *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*. pp96-103, Edmonton, Alberta, Canada.

C. D. Manning and H. Schütze. 1999. Foundations of Statistical Natural Language Processing. MIT Press. Boston MA.

H. Meng, W. K. Lo, B. Chen and K. Tang. 2001. Generating Phonetic Cognates to Handle Named Entities in English-Chinese Cross-Language Spoken Document Retrieval. *Proceedings of the ASRU-2001*, Trento, Italy, December.2001

D. Marcu and W. Wong. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. *Proceedings of EMNLP-2002,* Philadelphia, PA, 2002

F. J. Och, C. Tillmann, and H. Ney. Improved Alignment Models for Statistical Machine Translation. pp. 20-28; Proc. of the Joint Conf. of Empirical Methods in Natural Language Processing and Very Large Corpora; University of Maryland, College Park, MD, June 1999.

Y. Qu, and G. Grefenstette. Finding Ideographic Representations of Japanese Names Written in Latin Script via Language Identification and Corpus Validation. ACL 2004: 183-190

P. Virga and S. Khudanpur. 2003. Transliteration of Proper Names in Cross-Lingual Information Retrieval. *Proceedings of the ACL-2003 Workshop on Multi-lingual Named Entity Recognition* Japan. July 2003.

S. Vogel, Y. Zhang, F. Huang, A. Tribble, A. Venogupal, B. Zhao and A. Waibel. The CMU Statistical Translation System, *Proceedings of MT Summit IX* New Orleans, LA, USA, September 2003

D. Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. Computational Linguistics 23(3):377-404, September 1997.

Extracting Personal Names from Email: Applying Named Entity Recognition to Informal Text

Einat Minkov and **Richard C. Wang**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15217
{einat,rcwang}@cs.cmu.edu

William W. Cohen
Ctr for Automated Learning & Discovery
Carnegie Mellon University
Pittsburgh, PA 15217
wcohen@cs.cmu.edu

Abstract

There has been little prior work on Named Entity Recognition for ”informal” documents like email. We present two methods for improving performance of person name recognizers for email: email-specific structural features and a recall-enhancing method which exploits name repetition across multiple documents.

1 Introduction

Named entity recognition (NER), the identification of entity names in free text, is a well-studied problem. In most previous work, NER has been applied to news articles (e.g., (Bikel et al., 1999; McCallum and Li, 2003)), scientific articles (e.g., (Craven and Kumlien, 1999; Bunescu and Mooney, 2004)), or web pages (e.g., (Freitag, 1998)). These genres of text share two important properties: documents are written for a fairly broad audience, and writers take care in preparing documents. Important genres that do *not* share these properties include instant messaging logs, newsgroup postings and email messages. We refer to these genres as “informal” text.

Informal text is harder to process automatically. Informal documents do not obey strict grammatical conventions. They contain grammatical and spelling errors. Further, since the audience is more restricted, informal documents often use group- and task-specific abbreviations and are not self-contained. Because of these differences, existing NER methods may require modifications to perform well on informal text.

In this paper, we investigate NER for informal text with an experimental study of the problem of recognizing personal names in email—a task that is both useful and non-trivial. An application of interest is corpus anonymization. Automatic or semi-automatic email anonymization should allow using large amounts of informal text for research purposes, for example, of medical files. Person-name extraction and other NER tasks are helpful for automatic processing of informal text for a large variety of applications (Culotta et al., 2004; Cohen et al., 2005).

We first present four corpora of email text, annotated with personal names, each roughly comparable in size to the MUC-6 corpus[1]. We experimentally evaluate the performance of conditional random fields (CRF) (Lafferty et al., 2001), a state-of-the art machine-learning based NER methods on these corpora. We then turn to examine the special attributes of email text (vs. newswire) and suggest venues for improving extraction performance. One important observation is that email messages often include some structured, easy-to-recognize names, such as names within a header, names appearing in automatically-generated phrases, as well as names in signature files or sign-offs. We therefore suggest a set of specialized *structural* features for email; these features are shown to significantly improve performance on our corpora.

We also present and evaluate a novel method for exploiting *repetition* of names in a test corpus. Techniques for exploiting name repetition within documents have been recently applied to newswire text

[1]Two of these are publicly available. The others can not be distributed due to privacy considerations.

(e.g., (Humphreys et al., 1998)), scientific abstracts (e.g., (Bunescu and Mooney, 2004)) and seminar announcements (Sutton and Mccallum, 2004); however, these techniques rely on either NP analysis or capitalization information to pre-identify candidate coreferent name mentions, features which are not reliable in email. Furthermore, we argue that name repetition in email should be inferred by examining *multiple documents* in a corpus, which is not common practice. We therefore present an alternative efficient scheme for increasing recall in email, using the whole corpus. This technique is shown to always improve recall substantially, and to almost always improve F1 performance.

2 Corpora

Two email corpora used in our experiments were extracted from the CSpace email corpus (Kraut et al., 2004), which contains email messages collected from a management course conducted at Carnegie Mellon University in 1997. In this course, MBA students, organized in teams of four to six members, ran simulated companies in different market scenarios. We believe this corpus to be quite similar to the work-oriented mail of employees of a small or medium-sized company. This text corpus contains three header fields: "From", "Subject", and "Time". *Mgmt-Game* is a subcorpora consisting of all emails written over a five-day period. In the experiments, the first day worth of email was used as a training set, the fourth for tuning and the fifth day as a test set. *Mgmt-Teams* forms another split of this data, where the training set contains messages between different teams than in the test set; hence in *Mgmt-Teams*, the person names appearing in the test set are generally different than those that appear in the training set.

The next two collections of email were extracted from the Enron corpus (Klimt and Yang, 2004). The first subset, *Enron-Meetings*, consists of messages in folders named "meetings" or "calendar"[2]. Most but not all of these messages are meeting-related. The second subset, *Enron-Random*, was formed by repeatedly sampling a user name (uniformly at random among 158 users), and then sampling an email from that user (uniformly at random).

Annotators were instructed to include nicknames and misspelled names, but exclude person names that are part of an email address and names that are part of a larger entity name like an organization or location (e.g., "David Tepper School of Business").

The sizes of the corpora are given in Table 1. We limited training size to be relatively small, reflecting a real-world scenario.

Corpus	# Documents			#Words	#Names
	Train	Tune	Test	x1000	
Mgmt-Teams	120	82	83	105	2,792
Mgmt-Game	120	216	264	140	2,993
Enron-Meetings	244	242	247	204	2,868
Enron-Random	89	82	83	286	5,059

Table 1: Summary of the corpora used in the experiments. The number of words and names refer to the whole annotated corpora.

3 Existing NER Methods

In our first set of experiments we apply CRF, a machine-learning based probabilistic approach to labeling sequences of examples, and evaluate it on the problem of extracting personal names from email. Learning reduces NER to the task of *tagging* (i.e., classifying) each word in a document. We use a set of five tags, corresponding to (1) a one-token entity, (2) the first token of a multi-token entity, (3) the last token of a multi-token entity, (4) any other token of a multi-token entity and (5) a token that is not part of an entity.

The sets of features used are presented in Table 2. All features are instantiated for the focus word, as well as for a window of 3 tokens to the left and to the right of the focus word. The *basic features* include the lower-case value of a token t, and its *capitalization pattern*, constructed by replacing all capital letters with the letter "X", all lower-case letters with "x", all digits with "9" and compressing runs of the same letter with a single letter. The *dictionary features* define various categories of words including common words, first names, last names [3] and "roster names" [4] (international names list, where first and

[2] with two exceptions: (a) six very large files were removed, and (b) one very large "calendar" folder was excluded.

[3] We used US Census' lists of the most common first and last names in the US, available from http://www.census.gov/genealogy/www/freqnames.html

[4] A dictionary of 16,623 student names across the country, obtained as part of the RosterFinder project (Sweeney, 2003)

Basic Features
t, lexical value, lowercase (binary form, e.g. $f(t$=”hello”)=1)
capitalization pattern of t (binary form, e.g. $f(t.cap$=x+)=1)
Dictionary Features
inCommon: t in common words dictionary
inFirst: t in first names dictionary
inLast: t in last names dictionary
inRoster: t in roster names dictionary
First: inFirst ∩ ¬isLast ∩ ¬inCommon
Last: ¬inFirst ∩ inLast ∩ ¬inCommon
Name: (First ∪ Last ∪ inRoster) ∩ ¬ inCommon
Title: t in a personal prefixes/suffixes dictionary
Org: t in organization suffixes dictionary
Loc: t in location suffixes dictionary
Email Features
t appears in the header
t appears in the “from” field
t is a probable “signoff” (≈ after two line breaks and near end of message)
t is part of an email address (regular expression)
does the word starts a new sentence (≈ capitalized after a period, question or exclamation mark)
t is a probable initial (X or X.)
t followed by the bigram ”and I”
t capitalized and followed by a pronoun within 15 tokens

Table 2: Feature sets

last names are mixed.) In addition, we constructed some composite dictionary features, as specified in Table 2: for example, a word that is in the first-name dictionary and is not in the common-words or last-name dictionaries is designated a ”sure first name”.

The common-words dictionary used consists of base forms, conjugations and plural forms of common English words, and a relatively small ad-hoc dictionary representing words especially common in email (e.g., ”email”, ”inbox”). We also use small manually created word dictionaries of prefixes and suffixes indicative of persons (e.g., ”mr”, ”jr”), locations (e.g., ”ave”) and organizations (e.g., ”inc”).

Email structure features: We perform a simplified document analysis of the email message and use this to construct some additional features. One is an indicator as to whether a token t is equal to some token in the ”from” field. Another indicates whether a token t in the email body is equal to some token appearing in the whole header. An indicator feature based on a regular expression is used to mark tokens that are part of a probable ”sign-off” (i.e., a name at the end of a message). Finally, since the annotation rules do not consider email addresses to be names, we added an indicator feature for tokens that are inside an email address.

l.2.mr	l.1.president
l.2.mrs	l.2.dr
l.1.jr	r.2.who
l.1.judge	r.2.jr
r.3.staff	l.3.by
l.2.ms	r.3.president
r.2.staff	l.3.by
r.1.family	l.3.rep
l.3.says	l.2.rep
r.3.reporter	r.1.administration

l.1.by	r.2.home
l.2.by	r.1.or
l.3.name	l.1.with
l.2.name	l.1.thanks
l.3.by	r.1.picked
r.3.his	l.3.meet
r.1.ps	r.1.started
r.3.home	r.1.told
r.1.and	l.2.prof
l.1.called	l.2.email

Figure 1: Predictive contexts for personal-name words for MUC-6 (left) and Mgmt-Game (right) corpora. A features is denoted by its direction comparing to the focus word (l/r), offset and lexical value.

We experimented with features derived from POS tags and NP-chunking of the email, but found the POS assignment too noisy to be useful. We did include some features based on approximate linguistic rules. One rule looks for capitalized words that are not common words and are followed by a pronoun within a distance of up to 15 tokens. (As an example, consider “Contact Puck tomorrow. *He* should be around.”). Another rule looks for words followed by the bigram “and I”. As is common for hand-coded NER rules, both these rules have high precision and low recall.

3.1 Email vs Newswire

In order to explore some of the differences between email and newswire NER problems, we stripped all header fields from the Mgmt-Game messages, and trained a model (using basic features only) from the resulting corpus of email bodies. Figure 1 shows the features most indicative of a token being part of a name in the models trained for the Mgmt-Game and MUC-6 corpora. To make the list easier to interpret, it includes only the features corresponding to tokens surrounding the focus word.

As one might expect, the important features from the MUC-6 dataset are mainly formal name titles such as ”mr”, ”mrs”, and ”jr”, as well as job titles and other pronominal modifiers such as ”president” and ”judge”. However, for the Mgmt-Game corpus, most of the important features are related to email-specific structure. For example, the features ”left.1.by” and “left.2.by” are often associated with a quoted excerpt from another email message, which in the Mgmt-Game corpus is often marked by mailers with text like ”Excerpts from mail: 7-

Sep-97 Re: paper deadline by Richard Wang". Similarly, features like "left.1.thanks" and "right.1.ps" indicate a "signoff" section of an email, as does "right.2.home" (which often indicates proximity to a home phone number appearing in a signature).

3.2 Experimental Results

We now turn to evaluate the usefulness of the feature sets described above. Table 3 gives entity-level F1 performance [5] for CRF trained models for all datasets, using the basic features alone (B); the basic and email-tailored features (B+E); the basic and dictionary features (B+D); and, all of the feature sets combined (B+D+E). All feature sets were tuned using the Mgmt-Game validation subset. The given results relate to previously unseen test sets.

Dataset	B	B+E	B+D	B+D+E
Mgmt-Teams	68.1	75.7	82.0	87.9
Mgmt-Game	79.2	84.2	90.7	91.9
Enron-Meetings	59.0	71.5	78.6	76.9
Enron-Random	68.1	70.2	72.9	76.2

Table 3: F1 entity-leavel performance for the sets of features, across all datasets, with CRF training.

The results show that the email-specific features are very informative. In addition, they show that the dictionary features are especially useful. This can be explained by the relatively weak contextual evidence in email. While dictionaries are useful in named entities extraction in general, they are in fact more essential when extracting names from email text, where many name mentions are part of headers, names lists etc. Finally, the results for the combined feature set are superior in most cases to any subset of the features.

Overall the level of performance using all features is encouraging, considering the limited training set size. Performance on Mgmt-Teams is somewhat lower than for Mgmt-Game mainly because (by design) there is less similarity between training and test sets with this split. Enron emails seem to be harder than Mgmt-Game emails, perhaps because they include fewer structured instances of names. Enron-Meetings emails also contain a number of constructs that were not encountered in the Mgmt-Game corpus, notably lists (e.g., of people attending a meeting), and also include many location and organization names, which are rare in Mgmt-Game. A larger set of dictionaries might improve performance for the Enron corpora.

[5] No credit awarded for partially correct entity boundaries.

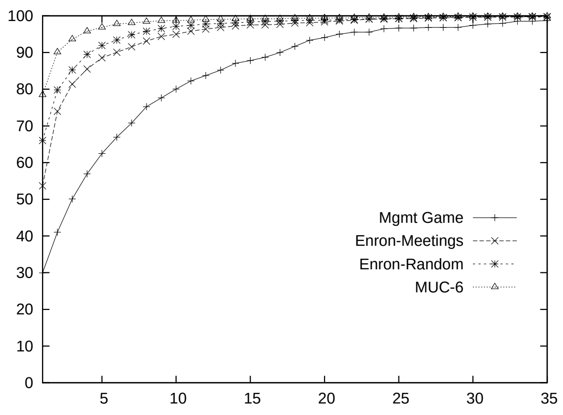


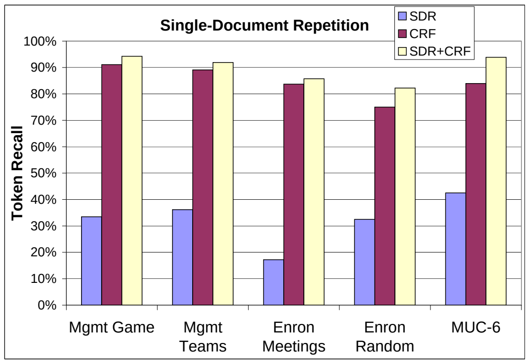
Figure 2: Cumulative percentage of person-name tokens w that appear in at most K distinct documents as a function of K.

4 Repetition of named entities in email

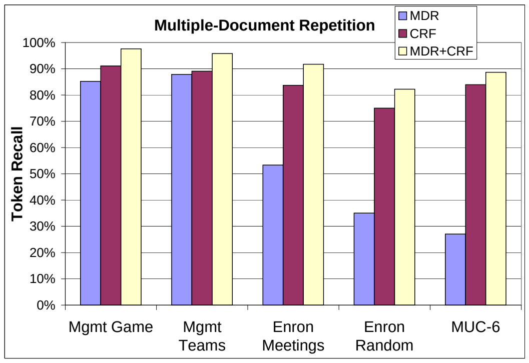
In the experiments described above, the extractors have high precision, but relatively low recall. This typical behavior suggests that some sort of recall-enhancing procedure might improve overall performance.

One family of recall-enhancing techniques are based on looking for multiple occurrences of names in a document, so that names which occur in ambiguous contexts will be more likely to be recognized. It is an intuitive assumption that the ways in which names repeat themselves in a corpus will be different in email and newswire text. In news stories, one would expect repetitions within a *single document* to be common, as a means for an author to establish a shared context with the reader. In an email corpus, one would expect names to repeat more frequently across the corpus, in *multiple documents*—at least when the email corpus is associated with a group that works together closely. In this section we support this conjecture with quantitative analysis.

In a first experiment, we plotted the percentage of person-name tokens w that appear in at most K distinct documents as a function of K. Figure 2 shows this function for the Mgmt-Game, MUC-6, Enron-Meetings, and Enron-Random datasets. There is a large separation between MUC-6 and Mgmt-Game, the most workgroup-oriented email corpus. In MUC-6, for instance, almost 80% of the



(a) SDR



(b) MDR

Figure 3: Upper bounds on recall and recall improvements associated with methods that look for terms that re-occur within a single document (SDR) or across multiple documents (MDR).

names appear in only a single document, while in Mgmt-Game, only 30% of the names appear in only a single document. At the other extreme, in MUC-6, only 1.3% of the names appear in 10 or more documents, while in Mgmt-Game, almost 20% do. The Enron-Random and Enron-Meetings datasets show distributions of names that are intermediate between Mgmt-Game and MUC-6.

As a second experiment, we implemented two very simple extraction rules. The *single document repetition* (SDR) rule marks every token that occurs more than once inside a single document as a name. Adding tokens marked by the SDR rule to the tokens marked by the learned extractor generates a new extractor, which we will denote SDR+CRF. Thus, the recall of SDR+CRF serves as an upper bound on the token recall[6] of any recall-enhancing method that improves the extractor by exploiting repetition within a single document. Analogously, the *multiple document repetition* (MDR) rule marks every token that occurs in more than one document as a name. Again, the token recall of MDR+CRF rule is an upper bound on the token recall of any recall-enhancing method that exploits token repetition across multiple documents.

The left bars in Figure 3 show the recall obtained by the SDR (top) and the MDR rule (bottom). The MDR rule has highest recall for the two Mgmt corpora, and lowest recall for the MUC-6 corpus. Conversely, for the SDR rule, the highest recall level obtained is for MUC-6. The middle bars show the token recall obtained by the CRF extractor, using all features. The right bars show the token recall of the SDR+CRF and MDR+CRF extractors. Comparing them to the other bars, we see that the maximal potential recall gain from a SDR-like method is on MUC-6. For MDR-like methods, there are large potential gains on the Mgmt corpora as well as on Enron-Meetings and Enron-Random to a lesser degree. This probably reflects the fact that the Enron corpora are from a larger and more weakly interacting set of users, compared to the Mgmt datasets.

These results demonstrate the importance of exploiting repetition of names across multiple documents for entity extraction from email.

5 Improving Recall With Inferred Dictionaries

Sequential learners of the sort used here classify tokens from each document independently; moreover, the classification of a word w is independent of the classification of other occurrences of w elsewhere in the document. That is, the fact that a word w has appeared somewhere in a context that clearly indicates that it is a name does not increase the probability that it will be classified as a name in other, more ambiguous contexts.

Recently, sequential learning methods have been extended to directly utilize information about name co-occurrence in learning the sequential classifier. This approach provides an elegant solution to modeling repetition within a single document. However, it requires identifying candidate related entities in advance, applying some heuristic. Thus, Bunescu &

[6]Token level recall is recall on the task of classifying tokens as inside or outside an entity name.

Mooney (2004) link between similar NPs (requiring their head to be identical), and Sutton and Mccallum (2004) connect pairs of identical capitalized words. Given that in email corpora capitalization patterns are not followed to a large extent, there is no adequate heuristic that would link candidate entities prior to extraction. Further, it is not clear if a collective classification approach can scale to modeling multiple-document repetition.

We suggest an alternative approach of recall-enhancing name matching, which is appropriate for email. Our approach has points of similarity to the methods described by Stevenson and Gaizauskas (2000), who suggest matching text against name dictionaries, filtering out names that are also common words or appear as non-names in high proportion in the training data. The approach described here is more systematic and general. In a nutshell, we suggest applying the *noisy* dictionary of predicted names over the test corpus, and use the approximate (predicted) name to non-name proportions over the test set itself to filter out ambiguous names. Therefore, our approach does not require large amount of annotated training data. It also does not require word distribution to be similar between train and test data. We will now describe our approach in detail.

5.1 Matching names from dictionary

First, we construct a dictionary comprised of all spans predicted as names by the learned model. For personal names, we suggest expanding this dictionary further, using a transformation scheme. Such a scheme would construct a family of possible variations of a name n: as an example, Figure 4 shows name variations created for the name span "Benjamin Brown Smith". Once a dictionary is formed, a single pass is made through the corpus, and every longest match to some name-variation is marked as a name[7]. It may be that a partial name span n_1 identified by the extractor is subsumed by the full name span n_2 identified by the dictionary-matching scheme. In this case, entity-level precision is increased, having corrected the entity's boundaries.

[7]Initials-only variants of a name, e.g., ”bs” in Figure 4 are marked as a name only if the “inSignoff” feature holds—i.e., if they appear near the end of a message in an apparent signature.

benjamin brown smith	benjamin-brown-s.	b. brown s.	bbs
benjamin-brown smith	benjamin-b. s.	b. b. smith	bs
benjamin brown-smith	benjamin-smith	b. brown-s.	
benjamin-brown-smith	benjamin smith	benjamin	
benjamin brown s.	b. brown smith	brown	
benjamin-b. smith	benjamin b. s.	smith	
benjamin b. smith	b. brown-smith	b. smith	
benjamin brown-s.	benjamin-s.	b. b. s	
benjamin-brown s.	benjamin s.	b. s.	

Figure 4: Names variants created from the name “Benjamin Brown Smith”

5.2 Dictionary-filtering schemes

The noisy dictionary-matching scheme is susceptible to false positives. That is, some words predicted by the extractor to be names are in fact non-names. Presumably, these non-names could be removed by simply eliminating low-confidence predictions of the extractor; however, ambiguous words –that are not exclusively personal names in the corpus– may need to be identified and removed as well. We note that ambiguity better be evaluated in the context of the corpus. For example, "Andrew" is a common first name, and may be confidently (and correctly) recognized as one by the extractor. However, in the Mgmt-Game corpus, "Andrew" is also the name of an email server, and most of the occurrences of this name in this corpus are *not* personal names. The high frequency of the word "Andrew" in the corpus, coupled with the fact that it is only sometimes a name, means that adding this word to the dictionary leads to a substantial drop in precision.

We therefore suggest a measure for filtering the dictionary. This measure combines two metrics. The first metric, *predicted frequency* (PF), estimates the degree to which a word appears to be used consistently as a name throughout the corpus:

$$PF(w) \equiv \frac{cpf(w)}{ctf(w)}$$

where $cpf(w)$ denotes the number of times that a word w is predicted as part of a name by the extractor, and $ctf(w)$ is the number of occurrences of the word w in the entire test corpus (we emphasize that estimating this statistic based on test data is valid, as it is fully automatic ”blind” procedure).

Predicted frequency does not assess the likely cost of adding a word to a dictionary: as noted above, ambiguous or false dictionary terms that occur frequently will degrade accuracy. A number of statistics could be used here; for instance, practitioners sometimes filter a large dictionary by simply discarding all words that occur more than k times in a test corpus. We elected to use the *inverse document frequency* (IDF) of w to measure word frequency:

$$IDF(w) \equiv \frac{\log(\frac{N+0.5}{df(w)})}{\log(N+1)}$$

Here $df(w)$ is the number of documents that contain a word w, and N is the total number of documents in the corpus. Inverse document frequency is often used in the field of information retrieval (Allan et al., 1998), and the formula above has the virtue of being scaled between 0 and 1 (like our PF metric) and of including some smoothing. In addition to bounding the cost of a dictionary entry, the IDF formula is in itself a sensible filter, since personal names will not appear as frequently as common English words.

The joint filter combines these two multiplicatively, with equal weights:

$$PF.IDF(w) : PF(w) \times IDF(w)$$

PF.IDF takes into consideration both the probability of a word being a name, and how common it is in the entire corpus. Words that get low PF.IDF scores are therefore either words that are highly ambiguous in the corpus (as derived from the extractors' predictions) or are common words, which were inaccurately predicted as names by the extractor.

In the MDR method of Figure 3, we imposed an artificial requirement that words must appear in more than one document. In the method described here, there is no such requirement: indeed, words that appear in a small number of documents are given higher weights, due to the IDF factor. Thus this approach exploits both single-document and multiple-document repetitions.

In a set of experiments that are not described here, the PF.IDF measure was found to be robust to parameter settings, and also preferable to its separate components in improving recall at minimal cost in precision. As described, the PF.IDF values per word range between 0 and 1. One can vary the threshold, under which a word is to be removed from the dictionary, to control the precision-recall trade-off. We tuned the PF.IDF threshold using the validation subsets, optimizing entity-level F1 (a threshold of 0.16 was found optimal).

In summary, our recall-enhancing strategy is as follows:

1. Learn an extractor E from the training corpus C_{train}.
2. Apply the extractor E to a test corpus C_{test} to assign a preliminary labeling.
3. Build a dictionary S_{θ_*} including the names n such that (a) n is extracted somewhere in the preliminary labeling of the test corpus, or is derived from an extracted name applying the name transformation scheme and (b) $PF.IDF(n) > \theta_*$.
4. Apply the dictionary-matching scheme of Section 5.1, using the dictionary S_{θ_*} to augment the preliminary labeling, and output the result.

5.3 Experiments with inferred dictionaries

Table 4 shows results using the method described above. We consider all of the email corpora and the CRF learner, trained with the full feature set. The results are given in terms of relative change, compared to the baseline results generated by the extractors ($score_{result}/score_{baseline} - 1$) and final value.

As expected, recall is always improved. Entity-level F1 is increased as well, as recall is increased more than precision is decreased. The largest improvements are for the Mgmt corpora —the two e-mail datasets shown to have the largest potential improvement from MDR-like methods in Figure 3. Recall improvements are more modest for the Enron datasets, as was anticipated by the MDR analysis. Another reason for the gap is that extractor baseline performance is lower for the Enron datasets, so that the Enron dictionaries are noisier.

As detailed in Section 2, the Mgmt-Teams dataset was constructed so that the names in the training and test set have only minimal overlap. The performance improvement on this dataset shows that repetition of mostly-novel names can be detected using our method. This technique is highly effective when names are novel, or dense, and is optimal when extractor baseline precision is relatively high.

Dataset	Precision	Recall	F1
Mgmt-Teams	-0.9% / 92.9	+8.5% / 89.8	+3.9% / 91.3
Mgmt-Game	-0.8% / 94.5	+8.4% / 96.2	+3.8% / 95.4
Enron-Meetings	-2.5% / 81.1	+4.7% / 74.9	+1.2% / 77.9
Enron-Random	-3.8% / 79.2	+4.9% / 74.3	+0.7% / 76.7

Table 4: Entity-level relative improvement and final result, applying name-matching on models trained with CRF and the full feature set (F1 baseline given in Table 3).

6 Conclusion

This work applies recently-developed sequential learning methods to the task of extraction of named entities from email. This problem is of interest as an example of NER from informal text—text that has been prepared quickly for a narrow audience.

We showed that informal text has different characteristics from formal text such as newswire. Analysis of the highly-weighted features selected by the learners showed that names in informal text have different (and less informative) types of contextual evidence. However, email also has some structural regularities which make it easier to extract personal names. We presented a detailed description of a set of features that address these regularities and significantly improve extraction performance on email.

In the second part of this paper, we analyzed the way in which names repeat in different types of corpora. We showed that repetitions within a single document are more common in newswire text, and that repetitions that span multiple documents are more common in email corpora. Additional analysis confirms that the potential gains in recall from exploiting multiple-document repetition is much higher than the potential gains from exploiting single-document repetition.

Based on this insight, we introduced a simple and effective method for exploiting multiple-document repetition to improve an extractor. One drawback of the recall-enhancing approach is that it requires the entire test set to be available: however, our test sets are of only moderate size (83 to 264 documents), and it is likely that a similar-size sample of unlabeled data would be available in many practical applications. The approach substantially improves recall and often improves F1 performance; furthermore, it can be easily used with any NER method.

Taken together, extraction performance is substantially improved by this approach. The improvements seem to be strongest for email corpora collected from closely interacting groups. On the Mgmt-Teams dataset, which was designed to reduce the value of memorizing specific names appearing in the training set, F1 performance is improved from 68.1% for the out-of-the-box system (or 82.0% for the dictionary-augmented system) to 91.3%. For the less difficult Mgmt-Game dataset, F1 performance is improved from 79.2% for an out-of-the-box CRF-based NER system (or 90.7% for a CRF-based system that uses several large dictionaries) to 95.4%. As future work, experiments should be expanded to include additional entity types and other types of informal text, such as blogs and forum postings.

References

J. Allan, J. Callan, W.B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. 1998. Inquery does battle with trec-6. In *TREC-6*.

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231.

R. Bunescu and R. J. Mooney. 2004. Relational markov networks for collective information extraction. In *ICML-2004 Workshop on Statistical Relational Learning*.

W. W. Cohen, E. Minkov, and A. Tomasic. 2005. Learning to undertand website update requests. In *IJCAI-05*.

M. Craven and J. Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB-99*.

A. Culotta, R. Bekkerman, and A. McCallum. 2004. Extracting social networks and contact information from email and the web. In *CEAS-04*.

D. Freitag. 1998. Information extraction from html: application of a general machine learning approach. In *AAAI-98*.

K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. 1998. Description of the LASIE-II system as used for MUC-7.

B. Klimt and Y. Yang. 2004. Introducing the Enron corpus. In *CEAS-04*.

R. E. Kraut, S. R. Fussell, F. J. Lerch, and J. A. Espinosa. 2004. Coordination in teams: evi-dence from a simulated management game. To appear in the Journal of Organizational Behavior.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-01*.

A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CoNLL-2003*.

M. Stevenson and R. Gaizauskas. 2000. Using corpus-derived names lists for named entities recognition. In *NAACL-2000*.

C. Sutton and A. Mccallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on Statistical Relational Learning*.

L. Sweeney. 2003. Finding lists of people on the web. Technical Report CMU-CS-03-168, CMU-ISRI-03-104. http://privacy.cs.cmu.edu/dataprivacy/ projects/rosterfinder/.

Matching Inconsistently Spelled Names in Automatic Speech Recognizer Output for Information Retrieval

Hema Raghavan and James Allan
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
{hema,allan}@cs.umass.edu

Abstract

Many proper names are spelled inconsistently in speech recognizer output, posing a problem for applications where locating mentions of named entities is critical. We model the distortion in the spelling of a name due to the speech recognizer as the effect of a noisy channel. The models follow the framework of the IBM translation models. The model is trained using a parallel text of closed caption and automatic speech recognition output. We also test a string edit distance based method. The effectiveness of these models is evaluated on a name query retrieval task. Our methods result in a 60% improvement in F1. We also demonstrate why the problem has not been critical in TREC and TDT tasks.

1 Introduction

Proper names are key to our understanding of topics in news. For example, to determine that a news story is on the 2004 elections in the United States, the words *President Bush*, *John Kerry* and *USA* are necessary features of the story. In other words, names of people, places and organizations are key entities of a news story. For many tasks, like in topic detection and tracking (TDT), the entities form an important feature for distinguishing topics from one another. For example, it is the people that distinguish stories on the 2004 election from stories on the 2000 U.S election. Names, especially rare and foreign ones are a problem for automatic speech recognition (ASR) systems as they are often out of vocabulary (OOV) i.e., they do not exist in the lexicon of the ASR system. An OOV word is replaced with the most similar word in the lexicon of the speech recognizer. Sometimes, even if a name is in the lexicon of the speech recognizer, it may have multiple spelling variants. The following is a sample ASR snippet from the TDT3 [1] corpus that demonstrates how the same entity may have different spellings even within the same snippet of ASR text.

...newspaper quotes **qaddafi** *is saying they'll turn them over but only if they're allowed ..leader moammar* **gadhafi** *says he doesn't want an international confrontation over the suspects in the..*

In this work, we aim to find methods by which to cluster or group names in ASR text. We evaluate a variety of techniques that range from a simple string-edit distance model to generative models using both intrinsic and extrinsic evaluations. We get statistically significant improvements in results for ad-hoc retrieval when the query is just the name of a person. We also explain why the problem of misspelled proper names in ASR has not been an issue in the TREC spoken document retrieval (SDR) track or in topic detection and tracking (TDT). We demonstrate how the problem would be of significance when the query is short, containing mainly names with little or no context.

[1] http:///www.ldc.upenn.edu/projects/tdt3/

2 Related Work

That names can be spelled differently is a problem that has been addressed by the database community in great detail. They found that the problem was rising in significance with the increasing interest in reconciling different databases. Differences in names due to spelling errors, spelling variants and transliteration errors have been dealt with by different kinds of approximate string matching techniques like Soundex, Phonix, and String Edit distance (James C. French, 1997; Zobel and Dart, 1996). The nature of the problem is identical when the domain consists of databases of documents but in order to apply techniques that were developed for names by the database community one would have to first detect names in the corpus, and then normalize them to some canonical form. This is the approach taken by Raghavan and Allan (Raghavan and Allan, 2004) who showed that normalizing names using Soundex codes resulted in a 10% improvement on the TDT3 Story Link Detection Task. They tested their method on newswire stories only. Their difficulty in applying Soundex to the ASR documents was that detecting names in ASR is too error prone for their methods to be useful (Miller et al., 2000).

Spoken document retrieval was a track at the TREC-6,7 and 8 (Voorhees and Harman, 1997; Voorhees and Harman, 1998; Voorhees and Harman, 1999) conferences. At the TREC-8 SDR track the conclusion was that ASR is not really an issue for ad hoc retrieval. However, the queries in those tracks were not centered on any entity. The TREC-8 proceedings also acknowledge that mean average precision dropped as named entity word error rate (NE-WER) increased. A typical speech recognizer has a lexicon of about 60K and for this size of a lexicon, about 10% of the person names are out of vocabulary (OOV).

The problem of alternate spellings of names has also been explored by the cross lingual information retrieval community (Virga and Khudanpur, 2003; AbdulJaleel and Larkey, 2003). The problem with names in machine translated text is quite similar to the problem with names in ASR text, except that the errors caused by a speech recognizer are often phonetic confusions, which is not necessarily the case for machine translation errors. Spelling errors of names in machine translated text are typically consistent. A given word in the source language always translates to the same word in the target language for a given machine translation system. As seen earlier, ASR systems do not exhibit such consistency.

Another problem that resembles the one we are addressing in this paper is that of spelling correction. Spelling correction has been tackled in several different ways (Durham et al., 1983), in some cases with the use of contextual cues (Golding and Roth, 1999) and in some cases it has been modeled as a "noisy channel problem" (Kernighan et al., 1990). The latter approach is similar to ours because we also approach the problem of spelling variations due to speech recognizer errors as analogous to the errors caused by a noisy channel. However, spelling correction methods must rectify human errors (typographic errors and common confusions) whereas speech recognizer errors are different.

Additionally, the argument that *Jon Smith* and *John Smythe* may genuinely be different people and should not be considered to be the same entity is more of a cross-document co-reference problem. The problem we are attempting to solve in this paper is one of grouping names that "sound like" each other together, without considering the problem of cross document co-reference. For example, the name *Lewinsky* has 199 occurrences in the TDT3 corpus, and also appears as *Lewinski* (1324 times), and *Lewenskey* (171 times). Most of these occurrences refer to *Monica Lewinsky*. The aim is to group all these variants together, without taking into consideration which ones refer to the same person. We then measure the effectiveness of our methods on various retrieval tasks.

Perhaps the most similar work from the point of view of the task is work in word spotting in audio output (Amir et al., 2001). The queries are single words and the task is to locate their mention in audio. The starting point in that work is however, a phonetic transcript of the audio signal and the emphasis is not on locating names. Our starting point is automatic speech recognizer output, and we aim to locate names in particular.

3 Our Approaches

In this section we explain the techniques by which we group names together. One method uses string edit distance to group names that are variants of each other. The other techniques are some of the possible generative models suitable to this task.

An **equivalence class** is defined as a group of names such that any two names in that class are variants of each other and such that there exist no two names from different equivalence classes that are variants of each other. An equivalence class is represented as a set of names enclosed in curly braces as {*name-1 name-2 ...*}

Four of our models are trained on a parallel text of ASR and manual transcripts (or closed caption depending on availability) in order to learn a probabilistic model of ASR errors. The parallel text consists of pairs of sentences: sentences from the ASR output and the corresponding manual transcripts. This is a common technique in machine translation for which the IBM translation models are popular methods (Brown et al., 1993).

As a convention, we use uppercase letters to denote ASR output and lowercase for manual transcriptions. Given an input of parallel text of ASR and manual transcriptions, the model learns a probabilistic dictionary. The dictionary contains pairs of closed caption and ASR words and the probability that the closed caption word is generated from a given word in ASR. Thus, the model might learn a high probability for *P(CAT|kate)*.

3.1 Overview of Methods

We generate equivalence classes of names by clustering a list of names. The algorithm draws links between pairs of words and then clusters the words into equivalence classes such that if a and b are linked and b and c are linked then *a, b* and c are in the same equivalence class. Links between words are generated in five different ways described below.

In the first of our methods we align manual transcripts and ASR sentences using the IBM translation model (Brown et al., 1993) to obtain a probabilistic dictionary. We give details of the translation model in section 3.2. Names are grouped such that if *P(CAT|kate)* is high (above some threshold) then there is a link between *CAT* and *kate*. This is called the *Simple Aligned* method. Some sample pairs of words obtained by this technique are shown in figure 1.

We can also ask a human to create a list of equivalence classes of names. We describe our method of obtaining such a list in section 4. This method is called the *Supervised* method.

african	AFRICA	albania	ALBANIAN
alex	ALEC	cardoso	CARDOZO
ann	ANNE	ching	CHIANG

Figure 1: Example of pairs of words obtained by Simple Aligned

Given a list of equivalence classes, pairs of names that go together can easily be generated such that for each pair, both words are obtained from the same equivalence class. In this way equivalence classes of names obtained from the Simple Aligned and Supervised methods can be used to create a list of pairs of names that form parallel text to train a character level machine translation model. We would expect this model to learn a high probability for similar sounding alphabets, e.g., a high probability for $P(C|k)$. Depending on where the training set of pairs of names for this method comes from, we get two possible systems. These are called the *Generative Unsupervised* method and *Generative Supervised* method respectively. Note that the Generative Unsupervised method is not completely unsupervised; we still need the parallel text of ASR and manual transcripts, but we don't need a human to do the added grouping of names into equivalence classes. A character level translation model helps us generalize better to unseen words.

We also grouped together names that differ by a string edit distance of one, giving a fifth system. In particular, we use the Levenshtein distance (Levenshtein, 1966), that is the number of insertions, deletions and substitutions needed to convert one string to the other. Many methods employed by the database community build on string edit distance. The method works well but has some disadvantages. Consider a user who types in a query containing a name such that the spelling, as typed by the user, never occurs in the corpus. To employ string edit distance, one would have to compare the query name against all the words in the vocabulary of the corpus to find the most similar strings. With a generative model, only the query needs to be expanded using the translation model, thereby speeding up the search process. The string edit distance model on the other hand, is completely unsupervised and needs no training in the form of parallel text. Both methods have their advantages and disadvantages, and the use of one method over the other is situation dependent.

3.2 Details

To learn alignments, translation probabilities, etc in the first method we used work that has been done in statistical machine translation (Brown et al., 1993), where the translation process is considered to be equivalent to a corruption of the source language text to the target language text due to a *noisy channel*. We can similarly consider that an ASR system corrupts the spelling of a name as a result of a noisy channel. To obtain the closed caption word c, of an ASR word a, we want to find the string for which the probability $P(c|a)$ is highest. This is modeled as

$$P(c|a) = \frac{P(c)P(a|c)}{P(a)} \tag{1}$$

For a given name a, since $P(a)$ is constant, the problem reduces to one of maximizing $P(c)P(a|c)$. $P(c)$ is called the language model. We need to model $P(a|c)$ as opposed to directly modeling $P(c|a)$ so that our model assigns more probability to well formed English names.

Given a pair of sentences (c, a), an alignment $\mathcal{A}(c, a)$ is defined as the mapping from the words in c to the words in a. If there are l closed caption words and m ASR words, there are 2^{lm} alignments in $\mathcal{A}(c, a)$. $l \in \mathcal{A}(c, a)$ can be denoted as a series $l_1^m = l_1, l_2 ... l_m$ where $l_j = i$ means that a word in position j of the ASR string is aligned with a word in position i of the closed caption string. Then $P(a|c)$ is computed as follows:

$$\begin{aligned} P(a|c) &= \sum_l P(a, l|c) \\ P(a, l|c) &= P(m|c) \prod_j^m P(l_j | l_1^{j-1}, a_1^{j-1}, m, e) \\ &\quad \times P(a_j | l_1^j, a_1^{j-1}, m, c) \end{aligned} \tag{2}$$

where a_j is a word in position j of the string a, and a_1^j is the series $a_1 ... a_j$. The model is generative in the following way: we first choose for each word in the closed caption string the number of ASR words that will be connected to it, then we pick the identity of those ASR words and finally we pick the actual positions that these words will occupy. There are five different IBM translation models (Brown et al., 1993). Models 3 and 4 build on the above equations, and also incorporate the notion of fertility. Fertility takes into account that a given word in closed caption may be omitted by an ASR system, or one word may result in two or more, like *Iraq* → *I ROCK* (This is a true example). The models are trained using Expectation Maximization. Further details are in the original paper (Brown et al., 1993).

The IBM models have shown good performance in machine translation, and especially so within certain families of languages, for example in translating between French and English or between Sinhalese and Tamil (Brown et al., 1993; Weerasinghe, 2004). Pairs of closed caption and ASR sentences or words (as the case may be) are akin to a pair of closely related languages.

For the Generative Unsupervised and Generative Supervised methods, we use the same models, but in this case the training set consists of pairs of words obtained from the ASR and closed caption text as opposed to sentences. In other words, the place of words in the previous case is taken by characters. Modeling fertility, etc, again fits very well in this case. For example the terminal character e is often dropped in ASR, and a single o in closed caption may result in a double o in ASR or vice versa.

4 Experimental Set Up

4.1 Corpora

For experiments in this paper we used the TREC-6 and TREC-7 SDR track data (Voorhees and Harman, 1998). We also used the TDT2 and TDT3 corpora. For TREC-6 we had the ASR output provided by NIST (WER 34%). The TREC-7 corpus consists of the output of the Dragon systems speech recognizer (WER 29.5%). For the TDT sources we had the ASR output of the BBN Byblos Speech recognizer provided by the LDC. NIST provides human generated transcripts for the TREC corpora and LDC provides closed caption quality transcripts with a WER of 14.5% for the TDT corpora. There are 3943, 23282, 1819 and 2866 ASR documents in the TDT2 TDT3, TREC-6 and TREC-7 corpora respectively.

4.2 Intrinsic Evaluation

The Paice evaluation (Paice, 1996) for stemming algorithms (algorithms that reduce a word to its morphological root), attempts to compare the equivalence classes generated by our methods with human judgments.

The Paice evaluation measures the performance of a stemmer based on its understemming and overstemming indices (UI and OI respectively). UI measures the total number of missed links between words and OI measures the total number of false alarm links. A perfect stemmer would have a UI and OI value of zero.

We obtained a list of names to be grouped into equivalence classes in the following way. We did not use a named entity tagger on the corpus because named entity taggers typically have very high word error rates for ASR text (Bikel et al., 1999). Instead we ran the Unix *spell* command on the corpus and used the list of rejected words as the list of names for the annotators to group into equivalence classes. These 296 OOV words are taken to correspond to the names in the corpus. We then obtained the set of ground-truth equivalence classes by a method similar to Paice.

A group of undergraduate students was hired. The list of names was provided to each student in a text editor in alphabetical order. The purpose as explained to them was to group together names that were alternate spellings of similar sounding names together. The student was instructed to go through the list systematically, and for each word to look at the previous 10 words, as well as the following 10 words to see if there were any other variants. If there was a word or a group where the current word was likely to fit in, they were asked to *cut* the word and *paste* it into the appropriate group. In this way, groups were created such that no word could belong to more than one group. The annotators were also asked to mark the words that were indeed names. Of the 296 OOV words, 292 were found to be actual names.

4.3 Extrinsic evaluation

In addition to the Paice evaluation we propose two extrinsic or task based evaluations for our methods. In the first task, given a name as a query, we aim to find all documents that have a mention of that name or any of its variants. In order to obtain queries and relevance judgments for this task we arbitrarily chose 35 groups of names from the ground-truth set of equivalence classes. The TDT3 corpus was chosen to be the test corpus for this task. Hence we eliminated those words that had no occurrence in the TDT3 corpus from the 35 groups of names giving a total of 76 names. Each of the 76 words formed a query. For each name query we consider all documents that contain a mention of any of the names in the equivalence class of the query as relevant to that query. In this way we obtained relevance judgments for the name query task. Some sample queries are shown in figure 2. We use $F1$ (harmonic mean of the precision and recall) as a measure of performance.

Query Equivalence class
1: {*christy christie*}
2: {*christina christine*}
3: {*toney toni*}
4: {*michelle michel mitchell*}
5: {*columbia colombia colombian*}

Figure 2: Some sample query equivalence classes

Our extrinsic evaluation is spoken document retrieval. The queries on the TREC-6 and TREC-7 corpora are standard TREC spoken document retrieval track queries. For the TDT2 corpus we use one randomly chosen document from each topic as the query. This document is like a long query with plenty of entities and plenty of contextual information. For the TDT3 corpus we use the topic descriptions as provided by the LDC as the queries. The LDC topic descriptions discuss the *events* that describe a topic and the key entities and locations involved in the event. These are representative of shorter queries, rich in entities. LDC has provided relevance judgments for both the TDT2 and TDT3 corpora. Mean average precision was used as the measure of evaluation.

4.4 Implementation Details

We use GIZA++ (Och and Ney, 2003) to train the machine translation system and the ISI ReWrite Decoder (ISI, 2001) to do the actual translations. The decoder takes as input the models learned by

GIZA++ and a sentence from the foreign language. It can output the top n translations of the input sentence. The ReWrite decoder can translate using IBM Model-3 or Model-4. We found Model 3 to have lower perplexity and hence chose it for our experiments. In order to build the language model $P(c)$, we used the CMU Language Modeling toolkit [2]. All retrieval experiments were performed using the LEMUR [3] toolkit, and using the traditional vector space model. In the traditional vector space model queries and documents are represented as vectors of words. Each word in the vector is weighted using a product of term frequency and inverse document frequency. The similarity between a query and a document is measured using the cosine of the angle between the query and document vectors.

The Simple Aligned and Generative Unsupervised methods require a parallel corpus of ASR and closed caption for training. For the name query task we used TDT2, TREC-6 and TREC-7 to train these methods and TDT3 as the test corpus.

The Supervised and Generative Supervised methods require a human to provide pairs of words that are variants of each other. We filtered out those words from the human generated list of equivalence classes that occurred exclusively in the test corpus and in no other corpus. This is equivalent to asking a human to group words in the training corpus. Similarly we trained the Simple Aligned and Generative Unsupervised models using ASR and closed caption text from all other sources except those in the test set.

The models were trained similarly for the SDR experiments. The models were tested on each of the four corpora in turn, and in each case they were trained on everything but the test corpus.

5 Results

5.1 Intrinsic Experiments

Table 1 shows how the different methods perform on the intrinsic evaluation. We also show the UI and OI values for methods that use string edit distances of 2, 3, 4 and 5. Note that the Supervised method is the ground truth for this evaluation, and hence it has a UI and OI value of zero. A string edit distance of 1 has the lowest OI value, meaning there are very few false alarms. Higher string edit distances have lower UI values, with an increase in OI. We will interpret the UI and OI values again after observing performance on the retrieval tasks, so as to interpret the impact of missed links and false alarm links for retrieval.

Method	UI	OI
Simple Aligned	0.236	0.004
Supervised	0	0
Gen Sup	0.393	0.023
Gen Uns	0.351	0.003
Str. Ed. (1)	0.229	0.000
Str. Ed. (2)	0.083	0.003
Str. Ed. (3)	0.039	0.001
Str. Ed. (4)	0.031	0.124
Str. Ed. (5)	0.023	0.336

Table 1: Understemming and Overstemming indices for each of the methods (lower is better)

5.2 Name Query Retrieval experiments

The results of our experiments on the name query task are given in table 2. We report both Macro and Micro averaged (averaged over the equivalence classes of the queries) F1 measures. They do not differ much since the equivalence classes have almost the same number (2-3) of names.

From table 2, all methods improve the baseline F1 score significantly (statistical significance measured using a two tailed t-test with a confidence of 95%). In general, the Simple Aligned, Generative Unsupervised and string edit distance methods are the best performing for this task. The string edit distance improves the baseline by over 60%. The Supervised method is also not as good as the other four of our methods as it does not generalize well to names that occur exclusively in the test set.

String edit distance performs very well on certain equivalence classes of names. For example, on the equivalence class {*Seigal, Segal, Siegal, Siegel*} the precision and recall are 100% each since all of the words in the equivalence class differ from each other by a string edit distance of one. In the case of the equivalence class {*Lewenskey Lewinski Lewinsky*}, the term *Lewenskey* has a string edit distance of 2 (greater than one) from the other two members,

[2]http://mi.eng.cam.ac.uk/prc14/toolkit_documentation.html
[3]http://www.cs.cmu.edu/lemur

Method	Micro avg Recall	Micro avg Precision	Micro F1	Macro avg Recall	Macro avg Precision	Macro F1
Baseline	0.401	1	0.573	0.400	1	0.571
Simple Aligned	0.632	0.933	**0.754**	0.608	0.925	**0.734**
Sup	0.477	0.961	**0.638**	0.463	0.960	**0.625**
Gen Sup	0.530	0.937	**0.677**	0.517	0.938	**0.667**
Gen Uns	0.590	0.921	**0.720**	0.576	0.913	**0.706**
Str. Ed	0.752	0.867	**0.806**	0.751	0.871	**0.807**

Table 2: Results on the Name Query Retrieval task

Lewinsky and *Lewinski*. The equivalence class of {*John Jon Joan*} has very low precision and recall. This is because both *John* and *Jon* differ by a string edit distance of one from so many other names in the corpus, such as *Jong*, resulting in lowered precision.

The Simple Aligned method fails on names it has not seen in the training set. However, for cases like {*Greensborough Greensboro*} the link between these two names is detected using the simple aligned method and by no other. The generative methods can detect variations in spelling due to similar sounding alphabets. For example it can detect the link between *Sydney* and *Sidney*. The generative models were also able to learn that c and k are substitutable for each other. Therefore these models could detect the links between the words in the equivalence class {*Katherine Kathryn Catherine*}.

The Simple Aligned model performs well on the extrinsic evaluations although it has a high OI value. The intrinsic evaluations use judgments by humans. The Simple Aligned method would conflate *Kofi* and *Copy* into one class if that was a genuine ASR error and the alignment was correct, but these two words would not be conflated into the same equivalence class by our annotators and would actually count as a false alarm on the intrinsic evaluations. Therefore, although the OI is high for the Simple Aligned Method, on closer examination we found that some of the false alarms were actually representative of ASR errors.

5.3 Spoken Document Retrieval

We now move on to discuss results on the SDR task. For TDT3 we got statistically significant improvements (an improvement in mean average precision from 0.715 to 0.757) over the baseline using string edit distance. On the remaining corpora we got little or no improvement by our methods. We proceed to explain why this is the case for each of the corpora.

The TREC-7 corpus has only 5 queries with a mention of a name resulting in hardly any gains overall. Similar was the case for TREC-6. Again in the case of the TDT2 corpus, since we used entire documents as stories, there are enough words in the query that a few recognition errors can be tolerated and therefore traditional retrieval is good for the task. There is evidence from previous TREC tracks (Voorhees and Harman, 1999) that shorter queries result in a decrease in retrieval performance and hence we see some improvements for TDT3. Besides, the TDT3 queries were rich in names.

We wanted to check how our methods performed on outputs of different ASR systems. Spoken document retrieval on the TREC-7 data with the output of Dragon systems, which has a word error rate of 29.5%, results in an improvement of 6% using the Simple Aligned method. The NIST-B2 system with a higher WER (46.6%) has an improvement in Mean Average Precision of 6.5%. Similarly with the CUHTK (WER 35.6%) and NIST-B1 (WER 33.8%) and Sheffield (WER 24.6 %) systems we obtained improvements of 1.6%, 0.39% and 0.05% respectively using the Simple Aligned method. Thus, with increasing WER, the named entity word error rate increases significantly, and therefore the benefits of our method are more apparent in such situations.

6 Discussion and Conclusions

We showed (both intrinsically and extrinsically) that string edit distance is an effective technique for locating name variants. We also developed a set of generative models and showed that they are almost as effective at name finding and document retrieval, but are probably more efficient than string edit distance. The generative models need to be trained on parallel text and therefore require human effort for training the models. The advantage of one method over the other is dependent on the size of the corpus and the availability of resources.

The problem has not been of significance in previous TREC tasks or in TDT, because we have always escaped the problem of misspelled names by virtue of the nature of those tasks. In the TREC tasks very few queries are centered on an entity. In all the TDT tasks, one is usually required to compare entire stories with each other. A story is long enough that there are enough words that are in the vocabulary (just like a very long query) or that are correctly recognized, that the ASR errors do not really matter. Therefore, the TDT tasks also do not suffer as a result of these ASR errors.

We can improve and apply our methods to other domains like Switchboard data (Godfrey et al., 1992). Our methods also generalize well across languages since there are no language specific techniques employed.

7 Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. In *Proceedings of the 12th CIKM conference*, pages 139–146. ACM Press.

Arnon Amir, Alon Efrat, and Savitha Srinivasan. 2001. Advances in phonetic word spotting. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 580–582, New York, NY, USA. ACM Press.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

P. F. Brown, Steven A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Lingustics*, 19(2):263–311.

Ivor Durham, David A. Lamb, and James B. Saxe. 1983. Spelling correction in user interfaces. *Commun. ACM*, 26(10):764–773.

J. Godfrey, E. Holiman, and J. McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the International Conference on Acoustics, Speech and Signa Processing pp. I-517-520, 1992*, pages 517–520.

Andrew R. Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

2001. ISI rewrite decoder, http://www.isi.edu/licensed-sw/rewrite-decoder/.

Allison L. Powell James C. French. 1997. Applications of approximate word matching in information retrieval. In *Proceedings of the Sixth CIKM Conference*.

Mark D. Kernighan, Kenneth W. Church, , and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90*, pages 205–210.

V. I. Levenshtein. 1966. Binary codes capable of correcing deletions,insertions and reversals. *Phs. Dokl.*, 6:707–710.

David Miller, Richard Schwartz, Ralph Weischedel, and Rebecca Stone. 2000. Named entity extraction from broadcast news.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Chris D. Paice. 1996. Method for evaluation of stemming algorithms based on error counting. *JASIS*, 47(8):632–649.

Hema Raghavan and James Allan. 2004. Using soundex codes for indexing names in asr documents. In *Proceedings of the HLT NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *Proceedings of the 26th ACM SIGIR conference*, pages 365–366. ACM Press.

E. M. Voorhees and D. K. Harman, editors. 1997. *The Sixth Text REtrieval Conference (TREC 6)*. NIST.

E. M. Voorhees and D. K. Harman, editors. 1998. *The Seventh Text REtrieval Conference (TREC 7)*. NIST.

E. M. Voorhees and D. K. Harman, editors. 1999. *The Eighth Text REtrieval Conference (TREC 8)*. NIST.

Ruvan Weerasinghe. 2004. A statistical machine translation approach to Sinhala Tamil language translation. In *SCALLA 2004*.

Justin Zobel and Philip W. Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *Proceedings of the 19th ACM SIGIR Conference,(Special Issue of the SIGIR Forum)*, pages 166–172.

Part-of-Speech Tagging using Virtual Evidence and Negative Training

Sheila M. Reynolds and **Jeff A. Bilmes**
Department of Electrical Engineering
University of Washington
Seattle, WA 98195-2500
{sheila,bilmes}@ee.washington.edu

Abstract

We present a part-of-speech tagger which introduces two new concepts: virtual evidence in the form of an "observed child" node, and negative training data to learn the conditional probabilities for the observed child. Associated with each word is a flexible feature-set which can include binary flags, neighboring words, etc. The conditional probability of *Tag* given *Word* + *Features* is implemented using a factored language-model with back-off to avoid data sparsity problems. This model remains within the framework of Dynamic Bayesian Networks (DBNs) and is conditionally-structured, but resolves the label bias problem inherent in the conditional Markov model (CMM).

1 Introduction

A common sequence-labeling task in natural language processing involves assigning a part-of-speech (POS) tag to each word in the input text. Previous authors have used numerous HMM-based models (Banko and Moore, 2004; Collins, 2002; Lee et al., 2000; Thede and Harper, 1999) and other types of networks including maximum entropy models (Ratnaparkhi, 1996), conditional Markov models (Klein and Manning, 2002; McCallum et al., 2000), conditional random fields (CRF) (Lafferty et al., 2001), and cyclic dependency networks (Toutanova et al., 2003). All of these models make use of varying amounts of contextual information. In this paper, we present a new model which remains within the well understood framework of Dynamic Bayesian Networks (DBNs), and we show that it produces state-of-the-art results when applied to the POS-tagging task. This new model is conditionally-structured and, through the use of virtual evidence (Pearl, 1988; Bilmes, 2004), resolves the explaining-away problems (often described as label or observation bias) inherent in the CMM.

This paper is organized as follows. In section 2 we discuss the differences between a hidden Markov model (HMM) and the corresponding conditional Markov model (CMM). In section 3 we describe our observed-child model (OCM), introducing the notion of virtual evidence, and providing an information-theoretic foundation for the use of negative training data. In section 4 we discuss our experiments and results, including a comparison of three simple first-order models and state-of-the-art results from our feature-rich second-order OCM.

For clarity, the comparisons and derivations in sections 2 and 3 are done for first-order models using a single binary feature. The same ideas are then generalized to a higher order model with more features (including adjacent words).

2 Generative vs. Conditional Models

In this section we discuss the tradeoffs between the generative hidden Markov model (HMM) and the conditional Markov model (CMM). For pedagogical reasons, the figures and equations are for first order models with a single word-feature.

The HMM shown in Figure 1 includes a single

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 459–466, Vancouver, October 2005. ©2005 Association for Computational Linguistics

feature (the binary flag *isCap*) in addition to the word itself. Each observation, $o_i = (w_i, f_i)$, is a word-feature pair. Let $\mathbf{o} = \{o_i\}$ be the observation sequence and $\mathbf{s} = \{s_i\}$ be the associated tag (state) sequence. The HMM[1] factorizes the joint probability distribution over these two sequences as:

$$P(\mathbf{s}, \mathbf{o}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)P(f_i|s_i)$$

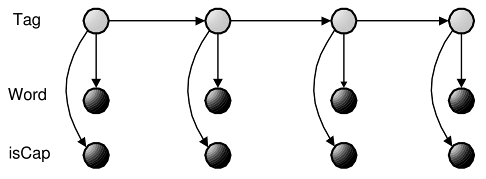


Figure 1: First order HMM.

A similar model often used for sequence labeling tasks is the conditional Markov model (CMM) which reverses the arrows between the words and the tags (Figure 2), and factorizes as:

$$P(\mathbf{s}, \mathbf{o}) = \prod_i P(s_i|s_{i-1}, w_i, f_i)P(w_i)P(f_i)$$

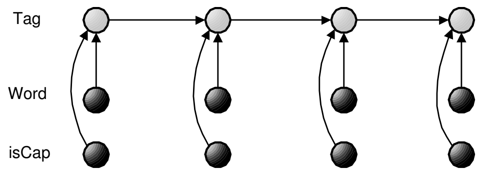


Figure 2: First order CMM.

Because the words and features are observed, this model does not require that we compute the probability of the evidence, $P(\mathbf{o})$, when finding the optimal tag sequence. The tag-sequence $\mathbf{s}$ which maximizes the joint probability $P(\mathbf{s}, \mathbf{o})$ is the same one that maximizes the *conditional* probability $P(\mathbf{s}|\mathbf{o})$. The CMM, therefore, does not require that we model the language, allowing us to focus on modeling the conditional probability of the tags given the words.

The HMM has its advantages as well, principally that it is easier to train than the CMM because it factorizes the joint probability into simpler components. The tables required for $P(s_i|s_{i-1})$ and $P(o_i|s_i)$ are significantly smaller than the one for $P(s_i|s_{i-1}, o_i)$ which may be difficult to estimate due to either data sparsity or normalization issues. One potential disadvantage of the HMM is that when it is trained using a maximum likelihood procedure, it is not necessarily encouraged to optimally classify tags due to its generative nature. One solution is to train the HMM using a discriminative procedure. Another option is to use entirely different models.

A key disadvantage of the CMM is that it makes critical statements about independence that the HMM does not: the converging arrows at each tag put the parent nodes (the previous tag and the current observation) into causal competition and as a result the model states that the previous tag is independent of the current observation. In other words, all states (tags) are independent of future observations (words). The CMM thus incorporates a strong directional bias which does not exist in the HMM.

One way to eliminate this bias is to use a CRF (Lafferty et al., 2001; McCallum, 2003), where factors over neighboring tags may use features from anywhere in the observation sequence. The CRF is discriminative and avoids label/observation bias by using a model that is constrained only in that the conditional distribution factorizes over an undirected Markov chain. However, most popular training procedures for a CRF are time-consuming and complex processes.

3 Using Virtual Evidence

Our goals in this work are to: 1) keep the discriminative nature of the CMM to the extent possible; 2) avoid label and observation bias issues; and 3) stay entirely within the DBN framework where training is relatively simple. We thus propose a new solution to the problem, which retains the discriminative conditional form of "tag given word" from the CMM, but avoids label bias by temporally linking adjacent tags in a new way. Specifically, we employ virtual evidence in the form of a binary observed child node, c_i, between adjacent tags (Figure 3) or a windowed sequence of tags. During decoding, this node will always be observed to be equal to 1 (one). Intuitively, this binary variable acts as an indicator of

[1]In this HMM, Word and isCap are independent given Tag, but this need not be true in general.

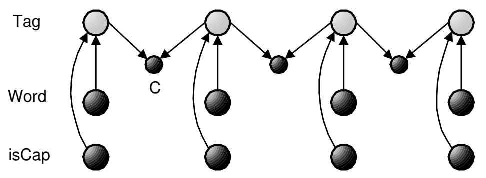


Figure 3: First order observed-child model (OCM) with the tags connected in pairs.

tag-pair consistency. When the tag pairs are consistent (as they are in real text), we should have a high conditional probability that $c_i = 1$; and when the tag pairs are not consistent, the conditional probability that $c_i = 1$ should be low. With this conditional distribution, observing $c_i = 1$ during decoding expresses a preference for consistent tag pairs.

The presence of this observed-child node results in a term in the factorization of the joint probability distribution that couples its parents:

$$P(\mathbf{c}, \mathbf{s}, \mathbf{o}) \propto \prod_i P(c_i | s_{i-1}, s_i) P(s_i | w_i, f_i)$$

where c_i is the observed-child node of tags s_{i-1} and s_i, and we omit the probability of the observations, $P(w_i, f_i)$ which do not affect the final choice of $\mathbf{s}$.

By the rules of d-separation (Pearl, 1988), the existence of c_i defined in this way means that the parents (the adjacent tags) are *not* conditionally independent given the child. This link between adjacent tags through an observed-child node allows for a probabilistic relationship to exist between the adjacent tags. Thus, future words can influence tags, which is not true for the CMM. Whether or not a relationship between tags will actually be learned, however, will critically depend on how the model is trained. In a graphical model, it is the *lack* of an edge that ensures some form of independence; the presence of an edge (or a path made up of two or more edges) does not necessarily ensure the reverse.

3.1 Training

The introduction of virtual evidence into a graphical model requires that careful thought be given to the training process. If we were to naïvely add $c_i = 1$ to all samples of the training data, the model would learn that c_i is constant rather than random, and therefore that it is independent of its parents, s_{i-1} and s_i. In other words, this naïvely-trained model would assume that $P(c_i = 1 | s_{i-1}, s_i) = 1 \; \forall \; (s_{i-1}, s_i)$, and when used to tag the sentences in the test-set (also labeled with $c_i = 1$), it would maximize this simplified joint probability in which the relationship between s_{i-1} and s_i has been lost:

$$P(\mathbf{c}, \mathbf{s}, \mathbf{o}) \propto \prod_i P(s_i | w_i, f_i)$$

In order to induce and thereby have the model *learn* the relationship between the adjacent tags s_{i-1} and s_i, the training has to be modified to include samples that are labeled with $c_i = 0$. The probability table $P(c_i = 1 | s_{i-1}, s_i)$ should favor common (consistent) tag-pairs with high probabilities, while discouraging rare tag-pairs with low probabilities.

Although all observations (in both training and test sets) are labeled with $c_i = 1$, we hypothesize an alternate set of observations labeled with $c_i = 0$. This alternate set will be the source of the negative training data [2]. It is a set of nonsensical sentences with the same distribution over individual tags, i.e. the same $P(s_i)$, but in this set adjacent tags are independent. We denote the total number of training samples by M. This is divided into positive training samples, M_1, and negative training samples, M_0, with $M_1 + M_0 = M$. The ratio of the amount of positive to negative training data should be the same as the ratio of our prior beliefs about tag-pair consistency, namely the ratio of $P(c_i = 1)$ to $P(c_i = 0)$. With no evidence to support that one is more likely than the other, one option is to use the strategy of "assuming the least" and use a maximum entropy prior, setting $M_0 = M_1$. More flexibly, we can define n to be the ratio of the two so that $M_0 = n \cdot M_1$.

Now we derive a method for training the conditional probability table $P(c_i | s_{i-1}, s_i)$ in terms of the pointwise mutual information between the adjacent tags s_{i-1} and s_i. We first rewrite the conditional probability (henceforth abbreviated as p) as:

$$p = P(c_i = 1 | s_{i-1}, s_i) = \frac{P(c_i = 1, s_{i-1}, s_i)}{P(s_{i-1}, s_i)}$$

If the probabilities are maximum likelihood (ML) estimates derived from counts on the training data, we can equivalently write:

$$p = \frac{\mathbf{N}(c_i = 1, s_{i-1}, s_i)}{\mathbf{N}(s_{i-1}, s_i)}$$

[2]This use of implied negative training data is similar to the "neighborhood" concept described in (Smith and Eisner, 2005)

where $\mathbf{N}(\cdot)$ is the count function.

Expanding the denominator into two terms:

$$p = \frac{\mathbf{N}(c_i = 1, s_{i-1}, s_i)}{\mathbf{N}(c_i = 1, s_{i-1}, s_i) + \mathbf{N}(c_i = 0, s_{i-1}, s_i)}$$

Without any negative training data (labeled with $c_i = 0$), this ratio would always evaluate to 1, and no probabilistic relationship between s_{i-1} and s_i would be learned.

From the start, we have implicitly postulated a relationship between adjacent tags. We now formally state two hypotheses: H_1 that there *is* a relationship between adjacent tags which can be described by some joint probability distribution $P(s_{i-1}, s_i)$, and the null hypothesis, H_0, that there is no such relationship, i.e. s_{i-1} and s_i are independent:

$$P_{H_1} = P(s_{i-1}, s_i)$$

$$P_{H_0} = P(s_{i-1})P(s_i)$$

Now we can express the counts as follows:

$$\mathbf{N}(c_i = 1, s_{i-1}, s_i) = M_1 \cdot P(s_{i-1}, s_i)$$

$$\mathbf{N}(c_i = 0, s_{i-1}, s_i) = M_0 \cdot P(s_{i-1})P(s_i)$$

where M_1 is the total number of tokens in the (positive) training data, and M_0 is the total number of tokens in the induced negative training data. We substitute M_0 with $n \cdot M_1$ for the reasons mentioned earlier, and simplify to obtain:

$$p = \frac{P(s_{i-1}, s_i)}{P(s_{i-1}, s_i) + nP(s_{i-1})P(s_i)}$$

which can be simplified to obtain:

$$p = \frac{1}{1 + n\left[\frac{P(s_{i-1},s_i)}{P(s_{i-1})P(s_i)}\right]^{-1}}$$

The ratio of probabilities in the denominator is the ratio used in computing the pointwise mutual information between s_{i-1} and s_i. This ratio, which we will call λ, is also the likelihood ratio between the two previously stated hypotheses. Finally, we write the conditional probability as a function of λ:

$$P(c_i = 1|s_{i-1}, s_i) = \frac{1}{1 + n\lambda^{-1}} = \frac{\lambda}{\lambda + n}$$

where $\lambda = \frac{P_{H_1}}{P_{H_0}} = \frac{P(s_{i-1}, s_i)}{P(s_{i-1})P(s_i)} = \frac{P(s_i|s_{i-1})}{P(s_i)}$

The conditional probability, $P(c_i = 1|s_{i-1}, s_i)$ is a mapping $g(\lambda)$ from $\lambda \in [0, \infty)$ to $p \in [0, 1)$.

Beginning with (Church and Hanks, 1989), numerous authors have used the pointwise mutual information between pairs of words to analyze word co-locations and associations. This ratio tells us whether s_{i-1} and s_i co-occur more or less often than would be expected by chance alone.

Consider, for example, the tags DT (determiner) and NN (noun), and the four possible ordered tag-pairs. The probabilities $P(s_i)$ and $P(s_i|s_{i-1})$ derived from the training data (see section 4.1), the likelihood ratio score λ, the conditional probability $p = P(c_i = 1|s_{i-1}, s_i)$, and the occurrence counts $\mathbf{N}$ are shown in Table 1. As expected, the sequence DT-NN (e.g. *the surplus*) occurs very often and gets a high score, while DT-DT (e.g. *this a*) and NN-DT (e.g. *surplus the*) occur infrequently and get low scores. The sequence NN-NN (e.g. *trade surplus*) gets a neutral score ($\lambda \approx 1$) indicating that if the preceding word is a noun, the likelihood that the current word is a noun is nearly equal to the likelihood that any randomly chosen word is a noun.

We present two methods for inducing the negative training counts that are required to train the conditional probability table for $P(c_i|s_{i-1}, s_i)$.

In the first method, we generate "nonsense" sentences by randomly scrambling each sentence in the training-set n times, using a uniform distribution over all possible permutations. This results in n negative training sentences for each positive training sentence and therefore $M_0 = n \cdot M_1$. Effectively, the ratio of priors on c_i is now:

$$\frac{P(c_i = 1)}{P(c_i = 0)} = \frac{M_1}{M_0} = \frac{1}{n}$$

The conditional probability table $P(c_i|s_{i-1}, s_i)$ is

s_{i-1}-s_i	$P(s_i)$	$P(s_i\|s_{i-1})$	λ	p	$\mathbf{N}$
DT-NN	0.129	0.4905	3.80	0.79	37301
NN-NN	0.129	0.1270	0.98	0.49	15571
NN-DT	0.080	0.0071	0.09	0.08	870
DT-DT	0.080	0.0018	0.02	0.02	134

Table 1: Sample likelihood ratio scores (λ), probabilities, p (for $n = 1$), and counts for four tag-pairs.

then trained using all $n+1$ versions of each sentence, thus inducing the desired dependence between s_{i-1} and s_i. The method of scrambling sentences n-times only approximates the theory described above because it is performed on a sentence-by-sentence basis rather than across the entire training set. Also, the resulting negative training data represents only n realizations of a random process, so the total number of samples may not be large enough to approximate the underlying distribution.

In the second method, rather than generate the negative training data in the form of scrambled sentences, we compute the negative-training counts directly, based on the positive unigram counts and the hypotheses presented in section 3.1. For example, the negative bigram counts are a function of the marginal probability of each tag, $P(s_i)$:

$$\mathbf{N}(c_i = 0, s_{i-1}, s_i) = nM_1 \cdot P(s_{i-1})P(s_i)$$

Negative unigram and trigram counts are computed in a similar fashion, and then the conditional probability table is derived as a smoothed back-off model directly from the combined sets of counts.

These two methods are conceptually similar but may exhibit subtle differences: one is randomizing at the sentence level while the other operates over the entire training set and does not have the same sensitivity to small values of n.

4 Experiments and Results

In this section we describe our experiments and the results obtained. Sections 4.1 and 4.2 describe the data sets and features. Section 4.3 presents comparisons between several simple models using just the tags, the words, and a single binary feature for each word. Section 4.4 presents results from a feature-rich second-order observed-child model in which tags are linked in groups of three.

All training of language models is done using the SRILM toolkit (Stolcke, 2002) with the FLM extensions (Bilmes and Kirchhoff, 2003), and the implementation and testing of the various graphical models is carried out with the help of the graphical models toolkit (GMTK) (Bilmes and Zweig, 2002).

4.1 Data Sets

The data used for these experiments is the Wall Street Journal data from Penn Treebank III (Marcus et al., 1994). We extracted tagged sentences from the parse trees and divided the data into training (sections 0-18), development (sections 19-21), and test (sections 22-24) sets as in (Toutanova et al., 2003). Except for the final results for the feature-rich model, all results are on the development set.

4.2 Features

The tagged sentences extracted from the Penn Treebank are pre-processed to generate appropriately-formatted training data for the SRILM toolkit, as well as the vocabulary and observation files to be used during testing.

The pre-processing includes building a dictionary based on the training data. All words containing uppercase letters are converted to lowercase before being written to the dictionary. Words that occur rarely are excluded from the dictionary and are instead mapped to a single out-of-vocabulary word. This is based on the idea from (Ratnaparkhi, 1996) that rare words in the training set are similar to unknown words in the test set, and can be used to learn how to tag the unknown words that will be encountered during testing. In this work, rare words are those that occurr fewer than 5 times. The dictionary also includes special *begin-sentence* and *end-sentence* words, as well as punctuation marks, resulting in a total of 10,824 words. A list of the 45 tags found in the training data is also created, and is similarly augmented with special *begin-sentence* and *end-sentence* tags, for a total of 47 distinct tags.

Each word has associated with it a set of features. During training, these features are used to learn a smoothed back-off model for $P(s_i|w_i, \mathbf{f}_i)$ (where $\mathbf{f}_i$ is a vector of features associated with word w_i).

The following five binary flags, taken from (Toutanova et al., 2003), are derived from the current word w_i and used as features :

- is-capitalized (refers to the first letter only);
- has-digits (word contains one or more digits);
- is-hyphenated (word contains '-');
- is-all-caps (*all* letters are capitalized);
- is-conjunction (true if is-all-caps, has-digits, and is-hyphenated are all true, for example *CFC-12* or *F/A-18*).

Prefixes and suffixes are also known to be informative and so we add a prefix-feature and a suffix-feature to our set. Previous work used all possible prefixes and suffixes ranging in length from 1 to k characters, with $k = 4$ (Ratnaparkhi, 1996), and $k = 10$ (Toutanova et al., 2003). This method results in very long lists of thousands of suffixes and prefixes. In this work, we instead analyzed the rare words in the training data to generate shorter lists of informative prefixes and suffixes, with lengths between 1 and 7 characters. Each prefix/suffix was scored based on the number of times it appeared with a particular tag, and all prefixes/suffixes that scored above 20 (an arbitrarily chosen threshold) were kept. This process resulted in two separate lists: one with 377 prefixes, and the other with 704 suffixes. Each word is then assigned a single prefix feature and a single suffix feature from these lists (which both include an entry for "unknown"). When assigning prefix and suffix features to the rare words (in the training data) or the unknown words (in the test data), we assume that the longest string is the most informative. (This may not necessarily be true: for example, although the suffix *ing* is certainly more informative than *g*, it is less clear whether *ulating* would be more or less informative than *ing*.)

We also include the two adjacent words as features of the current word. Our model provides great flexibility in the choice of features to be included in the current word's feature-set. This feature-set is not limited to binary flags and indeed can include anything that can be extracted from the observation sequence in the pre-processing stage. By using a smoothed back-off model, issues related to data-sparsity and over-fitting are avoided.

4.3 First Order Model Comparisons

In this section we compare results obtained from three first-order models: HMM, CMM, and OCM, using a Naïve Bayes (NB) model as a baseline. The Naïve Bayes model is a zeroth-order model with no connection between adjacent tags, while the first-order models connect adjacent tags in pairs. (Note that the HMM in this case is just a "temporal" NB since given the tag, the features are independent.) In these experiments, the only feature used is the is-capitalized flag (the most informative of the binary flags tested). The results are shown in Table 2.

The conditional probability tables (CPTs) for the CMM and the OCM were generated using the factored language model (FLM) extensions to the SRILM toolkit, wth generalized parallel backoff and Witten-Bell smoothing. (Modified Kneser-Ney smoothing could not be applied because some of the required low-order meta-counts needed by the discount estimator were zero.) The negative training data for the OCM was generated using the scramble method, with values of n as in the table. When no negative training data is used ($n = 0$), the CPT for the observed-child shows a very weak dependence on the specific tag-pair (s_{i-1}, s_i): the probability values in the tag-bigram model range only between 0.89 and 1. This weak dependence results in performance comparable to that of the Naïve Bayes model. That there is any dependence at all is due to the smoothing since $c_i = 0$ is never observed in the training data. With negative training data ($n = 4$), there is a much stronger dependence on the tag-pair, and the values for $P(c_i = 1|s_{i-1}, s_i)$ range between 0.0002 and 1.

model type	token accur.	known-w. accur.	unk.-w. accur.
Naïve Bayes	90.56%	93.83%	43.4%
OCM$_{n=0}$	90.89%	94.07%	45.2%
CMM	93.23%	95.69%	57.9%
OCM$_{n=1}$	93.94%	96.39%	58.6%
HMM	94.30%	96.53%	62.3%
OCM$_{n=4}$	94.42%	96.63%	62.7%

Table 2: Scores for first order models.

We found experimentally that the OCM reached peak performance with $n = 4$ and that for larger n the performance stayed relatively constant: the variation for values of n up to 14 was only 0.05%.

4.4 Feature-Rich Second-Order OCM

In this section we describe the results obtained from a more complex second order OCM with the additional word features described in section 4.2.

This model is illustrated in Figure 4 which, for clarity highlights the details only for one (tag,word) pair. The observed-child node, c_i, now has three parents: the tags s_{i-1}, s_i, and s_{i+1}. Each tag, s_i, in turn has $K + 1$ parents: the current word, w_i, and a set of K features (shown bundled together). The model switches between the two feature bundles as

model description	token accuracy	known-word accuracy	unknown-word accuracy
OCM-I, scramble, $n = 4$	96.39%	96.87%	89.5%
OCM-I, computed counts, $n = 4$	96.41%	96.90%	89.3%
OCM-I, computed counts, $n = 1$	96.41%	96.92%	89.0%
OCM-II, computed counts, $n = 1$	96.64%	97.12%	89.5%
OCM-II, as above, on test-set	96.77%	97.25%	90.0%

Table 3: Tagging accuracy using the feature-rich 2^{nd} order observed-child model.

illustrated, based on the current word. For known words, a small set of features is used, while a much larger set of features is used for unknown words. This switching increases the speed of the model at no cost: the additional features increase the tagging accuracy for unknown words but are redundant for known words.

This model factorizes the joint probability as:

$$P(\mathbf{c}, \mathbf{s}, \mathbf{o}) \propto \prod_i P(c_i | s_{i-1}, s_i, s_{i+1}) P(s_i | w_i, \mathbf{f}_i)$$

where $\mathbf{f}_i$ is the appropriate feature bundle for word w_i, depending on whether w_i is known or unknown.

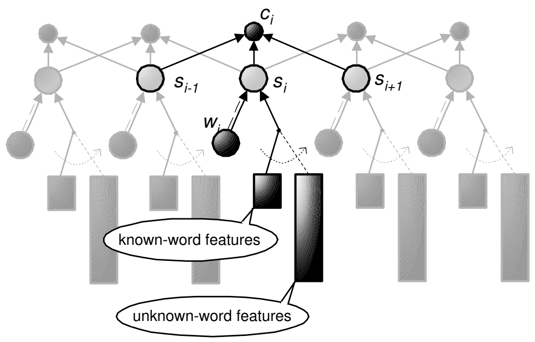


Figure 4: Second order OCM with tags connected in triples and switching sets of word features.

Two sets of experiments were performed using two models, which we will refer to as OCM-I and OCM-II. Both of these are second order models (connecting tags in triples), but with different sets of features. In model OCM-I, the only feature used for known words is the is-capitalized flag used in section 4.3. The unknown words use a total of seven features: suffix, prefix, and all five of the binary flags described in section 4.2. Model OCM-II adds the adjacent words (w_{i-1} and w_{i+1}) to the feature-set for both known and unknown words.

As seen above, the model factorizes the joint probability into two conditional probability terms. Each of these CPTs is implemented as a smoothed, factored-language back-off model.

The observed-child CPT uses generalized backoff, combining at run-time the results of backing off from each of the three parents if the specific tag-triple is not found in the table. The tag CPT uses linear backoff, dropping the adjacent words first. The backoff order for the other features was chosen based on experiments to determine the relative information content of each feature. This resulted in the following backoff order: prefix, has-digit, is-conjunction, is-all-caps, is-hyphenated, suffix, is-capitalized, word (where the least informative feature, prefix, is the first feature to be dropped).

Results from these experiments are shown in Table 3. Except for the last line, which reports results on the test set, all results are on the development set. The first three lines show results obtained from OCM-I (without adjacent word features). The two methods of generating negative training data yield nearly identical results, showing that they are comparable. Comparing rows 2 and 3 in the table we see that the computed-counts method is relatively insensitive to the value of n (for $n \geq 1$).

OCM-II, which uses the adjacent words as features for both known and unknown words further improves overall accuracy, and produces state-of-the-art results. The token-level accuracy result obtained from the OCM-II model on the development set (96.64%) can be directly compared to an accuracy of 96.57% reported in (Toutanova et al., 2003) for a cyclic dependency network using similar word features and the same three tag context.

5 Conclusions

In this paper, we have introduced two new concepts to the problem of part-of-speech tagging: virtual evidence and negative training data. We have moreover shown that this new model can produce state-of-the-art results on this NLP task with appropriately chosen features. The model stays entirely within the mathematically formal language of Bayesian networks, and even though it is conditional in nature, the model does not suffer from label or observation (or directional) bias. Staying within this framework has other advantages as well, including that the training procedures remain within the relatively simple maximum likelihood framework, albeit with appropriate smoothing. We believe that this model holds great promise for other NLP tasks as well as in other applications of machine-learning such as computational biology. In particular the way it factorizes the joint probability into a "horizontal" component which connects various nodes to the virtual-evidence node, and a "vertical" component (used here to link a tag to a set of observations), provides great simplicity, flexibility, and power.

6 Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments. Sheila Reynolds is supported by an NDSEG fellowship.

References

Michele Banko and Robert C. Moore. 2004. Part of Speech Tagging in Context. *Proceedings of COLING.*

Jeff Bilmes. 2004. On Soft Evidence in Bayesian Networks. Tech. Rep. UWEETR-2004-0016, U. Washington Dept. of Electrical Engineering, 2004.

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. *Proceedings of HLT-NAACL: Short Papers*, 4-6.

Jeff Bilmes and Geoffrey Zweig. 2002. The graphical models toolkit: An open source software system for speech and time-series processing. *Proceedings of ICASSP*, vol4, 3916-3919.

Kenneth W. Church and Patrick Hanks. 1989. Word Association Norms, Mutual Information, and Lexicography. *Proceedings of ACL*, 76-83.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. *Proc. EMNLP.*

Dan Klein and Christopher D. Manning. 2002. Conditional Structure versus Conditional Estimation in NLP Models. *Proceedings of EMNLP*, 9-16.

John Lafferty, Andrew McCallum and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of ICML*, 282-289.

Sang-Zoo Lee, Jun-ichi Tsujii and Hae-Chang Rim. 2000. Part-of-Speech Tagging Based on Hidden Markov Model Assuming Joint Independence. *Proceedings of 38th ACL*, 263-269.

Mitchell P. Marcus, Beatrice Santorini and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313-330.

Andrew McCallum. 2003. Efficiently Inducing Features of Conditional Random Fields. *Proceedings of UAI.*

Andrew McCallum, Dayne Freitag and Fernando Pereira. 2000. Maximum-Entropy Markov Models for Information Extraction and Segmentation. *Proc. 17th International Conf. on Machine Learning*, 591-598.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. *EMNLP 1*, 133-142.

Noah A. Smith and Jason Eisner 2005. Contrastive Estimation: Training Log-Linear Models on Unlabeled Data. *Proceedings of ACL.*

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. *Proc. ICASSP*, vol 2, 901-904.

Scott M. Thede and Mary P. Harper. 1999. A Second-Order Hidden Markov Model for Part-of-Speech Tagging. *Proceedings of 37th ACL*, 175-182.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. *Proceedings of HLT-NAACL*, 252-259.

Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data

Yoshimasa Tsuruoka[12] **and Jun'ichi Tsujii**[231]
[1] CREST, JST (Japan Science and Technology Corporation)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012 Japan
[2] Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Japan
[3] School of Informatics, University of Manchester
POBox 88, Sackville St, MANCHESTER M60 1QD, UK
{tsuruoka,tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper presents a bidirectional inference algorithm for sequence labeling problems such as part-of-speech tagging, named entity recognition and text chunking. The algorithm can enumerate all possible decomposition structures and find the highest probability sequence together with the corresponding decomposition structure in polynomial time. We also present an efficient decoding algorithm based on the easiest-first strategy, which gives comparably good performance to full bidirectional inference with significantly lower computational cost. Experimental results of part-of-speech tagging and text chunking show that the proposed bidirectional inference methods consistently outperform unidirectional inference methods and bidirectional MEMMs give comparable performance to that achieved by state-of-the-art learning algorithms including kernel support vector machines.

1 Introduction

The task of labeling sequence data such as part-of-speech (POS) tagging, chunking (shallow parsing) and named entity recognition is one of the most important tasks in natural language processing.

Conditional random fields (CRFs) (Lafferty et al., 2001) have recently attracted much attention because they are free from so-called label bias problems which reportedly degrade the performance of sequential classification approaches like maximum entropy markov models (MEMMs).

Although sequential classification approaches could suffer from label bias problems, they have several advantages over CRFs. One is the efficiency of training. CRFs need to perform dynamic programming over the whole sentence in order to compute feature expectations in each iteration of numerical optimization. Training, for instance, second-order CRFs using a rich set of features can require prohibitive computational resources. Max-margin methods for structured data share problems of computational cost (Altun et al., 2003).

Another advantage is that one can employ a variety of machine learning algorithms as the local classifier. There is huge amount of work about developing classification algorithms that have high generalization performance in the machine learning community. Being able to incorporate such state-of-the-art machine learning algorithms is important. Indeed, sequential classification approaches with kernel support vector machines offer competitive performance in POS tagging and chunking (Gimenez and Marquez, 2003; Kudo and Matsumoto, 2001).

One obvious way to improve the performance of sequential classification approaches is to enrich the information that the local classifiers can use. In standard decomposition techniques, the local classifiers cannot use the information about future tags (e.g. the right-side tags in left-to-right decoding), which would be helpful in predicting the tag of the target word. To make use of the information about future tags, Toutanova et al. proposed a tagging algorithm based on bidirectional dependency networks

(Toutanova et al., 2003) and achieved the best accuracy on POS tagging on the Wall Street Journal corpus. As they pointed out in their paper, however, their method potentially suffers from "collusion" effects which make the model lock onto conditionally consistent but jointly unlikely sequences. In their modeling, the local classifiers can always use the information about future tags, but that could cause a double-counting effect of tag information.

In this paper we propose an alternative way of making use of future tags. Our inference method considers all possible ways of decomposition and chooses the "best" decomposition, so the information about future tags is used only in appropriate situations. We also present a deterministic version of the inference method and show their effectiveness with experiments of English POS tagging and chunking, using standard evaluation sets.

2 Bidirectional Inference

The task of labeling sequence data is to find the sequence of tags $t_1...t_n$ that maximizes the following probability given the observation $o = o_1...o_n$

$$P(t_1...t_n|o). \tag{1}$$

Observations are typically words and their lexical features in the task of POS tagging. Sequential classification approaches decompose the probability as follows,

$$P(t_1...t_n|o) = \prod_{i=1}^{n} p(t_i|t_1...t_{i-1}o). \tag{2}$$

This is the left-to-right decomposition. If we make a first-order markov assumption, the equation becomes

$$P(t_1...t_n|o) = \prod_{i=1}^{n} p(t_i|t_{i-1}o). \tag{3}$$

Then we can employ a probabilistic classifier trained with the preceding tag and observations in order to obtain $p(t_i|t_{i-1}o)$ for local classification. A common choice for the local probabilistic classifier is maximum entropy classifiers (Berger et al., 1996). The best tag sequence can be efficiently computed by using a Viterbi decoding algorithm in polynomial time.

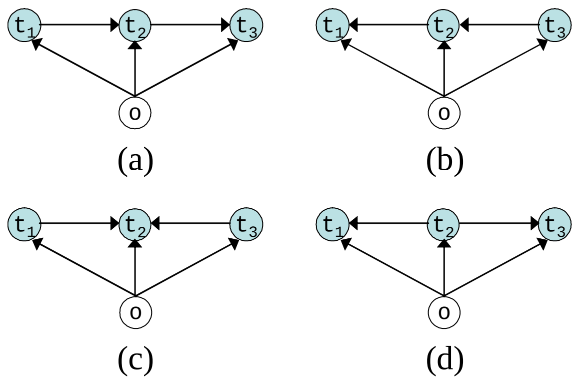


Figure 1: Different structures for decomposition.

The right-to-left decomposition is

$$P(t_1...t_n|o) = \prod_{i=1}^{n} p(t_i|t_{i+1}o). \tag{4}$$

These two ways of decomposition are widely used in various tagging problems in natural language processing. The issue with such decompositions is that you have only the information about the preceding (or following) tags when performing local classification.

From the viewpoint of local classification, we want to give the classifier as much information as possible because the information about neighboring tags is useful in general.

As an example, consider the situation where we are going to annotate a three-word sentence with part-of-speech tags. Figure 1 shows the four possible ways of decomposition. They correspond to the following equations:

$$(a)\ P(t_1...t_3|o) = P(t_1|o)P(t_2|t_1o)P(t_3|t_2o) \tag{5}$$

$$(b)\ P(t_1...t_3|o) = P(t_3|o)P(t_2|t_3o)P(t_1|t_2o) \tag{6}$$

$$(c)\ P(t_1...t_3|o) = P(t_1|o)P(t_3|o)P(t_2|t_3t_1o) \tag{7}$$

$$(d)\ P(t_1...t_3|o) = P(t_2|o)P(t_1|t_2o)P(t_3|t_2o) \tag{8}$$

(a) and (b) are the standard left-to-right and right-to-left decompositions. Notice that in decomposition (c), the local classifier can use the information about the tags on both sides when deciding t_2. If, for example, the second word is difficult to tag (e.g. an unknown word), we might as well take the decomposition structure (c) because the local classifier can use rich information when deciding the tag of the most difficult word. In general if we have an n-word sentence and adopt a first-order markov assumption, we have 2^{n-1} possible ways of decomposition because each of the $n-1$ edges in the corresponding graph has two directions (left-to-right or right-to-left).

Our bidirectional inference method is to consider all possible decomposition structures and choose the "best" structure and tag sequence. We will show in the next section that this is actually possible in polynomial time by dynamic programming.

As for the training, let us look at the equations of four different decompositions above. You can notice that there are only four types of local conditional probabilities: $P(t_i|t_{i-1}o)$, $P(t_i|t_{i+1}o)$, $P(t_i|t_{i-1}t_{i+1}o)$, and $P(t_i|o)$.

This means that if we have these four types of local classifiers, we can consider any decomposition structures in the decoding stage. These local classifiers can be obtained by training with corresponding neighboring tag information. Training the first two types of classifiers is exactly the same as the training of popular left-to-right and right-to-left sequential classification models respectively.

If we take a second-order markov assumption, we need to train 16 types of local classifiers because each of the four neighboring tags of a classification target has two possibilities of availability. In general, if we take a k-th order markov assumption, we need to train 2^{2k} types of local classifies.

2.1 Polynomial Time Inference

This section describes an algorithm to find the decomposition structure and tag sequence that give the highest probability. The algorithm for the first-order case is an adaptation of the algorithm for decoding the best sequence on a bidirectional dependency network introduced by (Toutanova et al., 2003), which originates from the Viterbi decoding algorithm for second-order markov models.

Figure 2 shows a polynomial time decoding algorithm for our bidirectional inference. It enumerates all possible decomposition structures and tag sequences by recursive function calls, and finds the highest probability sequence. Polynomial time is achieved by caching. Note that for each local classification, the function chooses the appropriate local classifier by taking into account the directions of the adjacent edges of the classification target.

```
function bestScore()
{
  return bestScoreSub(n+2, $\langle end, end, end \rangle$, $\langle L, L \rangle$);
}

function bestScoreSub(i+1, $\langle t_{i-1}, t_i, t_{i+1} \rangle$, $\langle d_{i-1}, d_i \rangle$)
{
  // memorization
  if (cached(i+1, $\langle t_{i-1}, t_i, t_{i+1} \rangle$, $\langle d_{i-1}, d_i \rangle$))
    return cache(i+1, $\langle t_{i-1}, t_i, t_{i+1} \rangle$, $\langle d_{i-1}, d_i \rangle$);
  // left boundary case
  if (i = -1)
    if ($\langle t_{i-1}, t_i, t_{i+1} \rangle$ = $\langle start, start, start \rangle$)    return 1;
    else                                                                                  return 0;
  // recursive case
  $P$ = localClassification(i, $\langle t_{i-1}, t_i, t_{i+1} \rangle$, $\langle d_{i-1}, d_i \rangle$);
  return $\max_{d_{i-2}} \max_{t_{i-2}} P \times$
          bestScoreSub(i, $\langle t_{i-2}, t_{i-1}, t_i \rangle$, $\langle d_{i-2}, d_{i-1} \rangle$);
}

function localClassification(i, $\langle t_{i-1}, t_i, t_{i+1} \rangle$, $\langle d_{i-1}, d_i \rangle$)
{
  if ($d_{i-1} = L$ & $d_i = L$)    return $P(t_i|t_{i+1}, o)$;
  if ($d_{i-1} = L$ & $d_i = R$)    return $P(t_i|o)$;
  if ($d_{i-1} = R$ & $d_i = L$)    return $P(t_i|t_{i-1}t_{i+1}, o)$;
  if ($d_{i-1} = R$ & $d_i = R$)    return $P(t_i|t_{i-1}, o)$;
}
```

Figure 2: Pseudo-code for bidirectional inference for the first-order conditional markov models. d_i is the direction of the edge between t_i and t_{i+1}.

The second-order case is similar but slightly more complex. Figure 3 shows the algorithm. The recursive function needs to consider the directions of the four adjacent edges of the classification target, and maintain the directions of the two neighboring edges to enumerate all possible edge directions. In addition, the algorithm rules out cycles in the structure.

2.2 Decoding with the Easiest-First Strategy

We presented a polynomial time decoding algorithm in the previous section. However, polynomial time is not low enough in practice. Indeed, even the Viterbi decoding of second-order markov models for POS tagging is not practical unless some pruning method is involved. The computational cost of the bidirectional decoding algorithm presented in the previous section is, of course, larger than that because it enumerates all possible directions of the edges on top of the enumeration of possible tag sequences.

In this section we present a greedy version of the decoding method for bidirectional inference, which

```
function bestScore()
{
  return bestScoreSub(n+3, $\langle end, end, end, end, end\rangle$, $\langle L, L, L, L\rangle$, $\langle L, L\rangle$);
}

function bestScoreSub(i+2, $\langle t_{i-2}, t_{i-1}, t_i, t_{i+1} t_{i+2}\rangle$, $\langle d'_{i-1}, d_{i-1}, d_i, d'_{i+1}\rangle$, $\langle d_{i-2}, d'_i\rangle$)
{
  // to avoid cycles
  if ($d_{i-1}$ = $d_i$ & $d_i$ != $d'_i$)   return 0;
  // memorization
  if (cached(i+2, $\langle t_{i-2}, t_{i-1}, t_i, t_{i+1} t_{i+2}\rangle$, $\langle d'_{i-1}, d_{i-1}, d_i, d'_{i+1}\rangle$, $\langle d_{i-2}, d'_i\rangle$)
    return cache(i+2, $\langle t_{i-2}, t_{i-1}, t_i, t_{i+1} t_{i+2}\rangle$, $\langle d'_{i-1}, d_{i-1}, d_i, d'_{i+1}\rangle$, $\langle d_{i-2}, d'_i\rangle$);
  // left boundary case
  if (i = -2)
    if ($\langle t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}\rangle$ = $\langle start, start, start, start, start\rangle$)   return 1;
    else                                                                      return 0;
  // recursive case
  $P$ = localClassification(i, $\langle t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}\rangle$, $\langle d'_{i-1}, d_{i-1}, d_i, d'_{i+1}\rangle$);
  return $\max_{d'_{i-2}} \max_{d_{i-3}} \max_{t_{i-3}} P\times$ bestScoreSub(i+1, $\langle t_{i-3}, t_{i-2}, t_{i-1}, t_i t_{i+1}\rangle$, $\langle d'_{i-2}, d_{i-2}, d_{i-1}, d'_i\rangle$, $\langle d_{i-3}, d'_{i-1}\rangle$);
}
```

Figure 3: Pseudo-code for bidirectional inference for the second-order conditional markov models. d_i is the direction of the edge between t_i and t_{i+1}. d'_i is the direction of the edge between t_{i-1} and t_{i+1}. We omit the localClassification function because it is the obvious extension of that for the first-order case.

is extremely simple and significantly more efficient than full bidirectional decoding.

Instead of enumerating all possible decomposition structures, the algorithm determines the structure by adopting the easiest-first strategy. The whole decoding algorithm is given below.

1. Find the “easiest” word to tag.
2. Tag the word.
3. Go back to 1. until all the words are tagged.

We assume in this paper that the “easiest” word to tag is the word for which the classifier outputs the highest probability. In finding the easiest word, we use the appropriate local classifier according to the availability of the neighboring tags. Therefore, in the first iteration, we always use the local classifiers trained with no contextual tag information (i.e. $(P(t_i|o))$. Then, for example, if t_3 has been tagged in the first iteration in a three-word sentence, we use $P(t_2|t_3 o)$ to compute the probability for tagging t_2 in the second iteration (as in Figure 1 (b)).

A naive implementation of this algorithm requires $O(n^2)$ invocations of local classifiers, where n is the number of the words in the sentence, because we need to update the probabilities over the words at each iteration. However, a k-th order Markov assumption obviously allows us to skip most of the probability updates, resulting in $O(kn)$ invocations of local classifiers. This enables us to build a very efficient tagger.

3 Maximum Entropy Classifier

For local classifiers, we used a maximum entropy model which is a common choice for incorporating various types of features for classification problems in natural language processing (Berger et al., 1996).

Regularization is important in maximum entropy modeling to avoid overfitting to the training data. For this purpose, we use the maximum entropy modeling with inequality constraints (Kazama and Tsujii, 2003). The model gives equally good performance as the maximum entropy modeling with Gaussian priors (Chen and Rosenfeld, 1999), and the size of the resulting model is much smaller than that of Gaussian priors because most of the parameters become zero. This characteristic enables us to easily handle the model data and carry out quick decoding, which is convenient when we repetitively perform experiments. This modeling has one parameter to tune, which is called the *width factor*. We tuned this parameter using the development data in each type of experiments.

Current word	w_i	& t_i
Previous word	w_{i-1}	& t_i
Next word	w_{i+1}	& t_i
Bigram features	w_{i-1}, w_i	& t_i
	w_i, w_{i+1}	& t_i
Previous tag	t_{i-1}	& t_i
Tag two back	t_{i-2}	& t_i
Next tag	t_{i+1}	& t_i
Tag two ahead	t_{i+2}	& t_i
Tag Bigrams	t_{i-2}, t_{i-1}	& t_i
	t_{i-1}, t_{i+1}	& t_i
	t_{i+1}, t_{i+2}	& t_i
Tag Trigrams	$t_{i-2}, t_{i-1}, t_{i+1}$	& t_i
	$t_{i-1}, t_{i+1}, t_{i+2}$	& t_i
Tag 4-grams	$t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}$	& t_i
Tag/Word combination	t_{i-1}, w_i	& t_i
	t_{i+1}, w_i	& t_i
	t_{i-1}, t_{i+1}, w_i	& t_i
Prefix features	prefixes of w_i (up to length 10)	& t_i
Suffix features	suffixes of w_i (up to length 10)	& t_i
Lexical features	whether w_i has a hyphen	& t_i
	whether w_i has a number	& t_i
	whether w_i has a capital letter	& t_i
	whether w_i is all capital	& t_i

Table 1: Feature templates used in POS tagging experiments. Tags are parts-of-speech. Tag features are not necessarily used in all the models. For example, "next tag" features cannot be used in left-to-right models.

Method	Accuracy (%)	Speed (tokens/sec)
Left-to-right (Viterbi)	96.92	844
Right-to-left (Viterbi)	96.89	902
Dependency Networks	97.06	1,446
Easiest-last	96.58	2,360
Easiest-first	**97.13**	2,461
Full bidirectional	97.12	34

Table 2: POS tagging accuracy and speed on the development set.

Method	Accuracy (%)
Dep. Networks (Toutanova et al., 2003)	97.24
Perceptron (Collins, 2002)	97.11
SVM (Gimenez and Marquez, 2003)	97.05
HMM (Brants, 2000)	96.48
Easiest-first	97.10
Full Bidirectional	97.15

Table 3: POS tagging accuracy on the test set (Sections 22-24 of the WSJ, 5462 sentences).

4 Experiments

To evaluate the bidirectional inference methods presented in the previous sections, we ran experiments on POS tagging and text chunking with standard English data sets.

Although achieving the best accuracy is not the primary purpose of this paper, we explored useful feature sets and parameter setting by using development data in order to make the experiments realistic.

4.1 Part-of-speech tagging experiments

We split the Penn Treebank corpus (Marcus et al., 1994) into training, development and test sets as in (Collins, 2002). Sections 0-18 are used as the training set. Sections 19-21 are the development set, and sections 22-24 are used as the test set. All the experiments were carried out on the development set, except for the final accuracy report using the best setting.

For features, we basically adopted the feature set provided by (Toutanova et al., 2003) except for complex features such as crude company-name detection features because they are specific to the Penn Treebank and we could not find the exact implementation details. Table 1 lists the feature templates used in our experiments.

We tested the proposed bidirectional methods, conventional unidirectional methods and the bidirectional dependency network proposed by Toutanova (Toutanova et al., 2003) for comparison. [1]. All the models are second-order. Table 2 shows the accuracy and tagging speed on the development data [2]. Bidirectional inference methods clearly outperformed unidirectional methods. Note that the easiest-first decoding method achieves equally good performance as full bidirectional inference. Table 2 also shows that the easiest-last strategy, where we select and tag the most difficult word at each iteration, is clearly a bad strategy.

An example of easiest-first decoding is given below:

[1]For dependency network and full bidirectional decoding, we conducted pruning because the computational cost was too large to perform exhaustive search. We pruned a tag candidate if the zero-th order probability of the candidate $P(t_i|o)$ was lower than one hundredth of the zero-th order probability of the most likely tag at the token.

[2]Tagging speed was measured on a server with an AMD Opteron 2.4GHz CPU.

The/DT/4 company/NN/7 had/VBD/11 sought/VBN/14 increases/NNS/13 totaling/VBG/12 $/$/2 80.3/CD/5 million/CD/8 ,/,/1 or/CC/6 22/CD/9 %/NN/10 ./././3

Each token represents Word/PoS/DecodingOrder. Typically, punctuations and articles are tagged first. Verbs are usually tagged in later stages because their tags are likely to be ambiguous.

We applied our bidirectional inference methods to the test data. The results are shown in Table 3. The table also summarizes the accuracies achieved by several other research efforts. The best accuracy is 97.24% achieved by bidirectional dependency networks (Toutanova et al., 2003) with a richer set of features that are carefully designed for the corpus. A perceptron algorithm gives 97.11% (Collins, 2002). Gimenez and Marquez achieve 97.05% with support vector machines (SVMs). This result indicates that bidirectional inference with maximum entropy modeling can achieve comparable performance to other state-of-the-art POS tagging methods.

4.2 Chunking Experiments

The task of chunking is to find non-recursive phrases in a sentence. For example, a chunker segments the sentence "He reckons the current account deficit will narrow to only 1.8 billion in September" into the following,

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only 1.8 billion] [PP in] [NP September] .

We can regard chunking as a tagging task by converting chunks into tags on tokens. There are several ways of representing text chunks (Sang and Veenstra, 1999). We tested the Start/End representation in addition to the popular IOB2 representation since local classifiers can have fine-grained information on the neighboring tags in the Start/End representation.

For training and testing, we used the data set provided for the CoNLL-2000 shared task. The training set consists of section 15-18 of the WSJ corpus, and the test set is section 20. In addition, we made the development set from section 21 [3].

We basically adopted the feature set provided in

Current word	w_i	& t_i
Previous word	w_{i-1}	& t_i
Word two back	w_{i-2}	& t_i
Next word	w_{i+1}	& t_i
Word two ahead	w_{i+2}	& t_i
Bigram features	w_{i-2}, w_{i-1}	& t_i
	w_{i-1}, w_i	& t_i
	w_i, w_{i+1}	& t_i
	w_{i+1}, w_{i+2}	& t_i
Current POS	p_i	& t_i
Previous POS	p_{i-1}	& t_i
POS two back	p_{i-2}	& t_i
Next POS	p_{i+1}	& t_i
POS two ahead	p_{i+2}	& t_i
Bigram POS features	p_{i-2}, p_{i-1}	& t_i
	p_{i-1}, p_i	& t_i
	p_i, p_{i+1}	& t_i
	p_{i+1}, p_{i+2}	& t_i
Trigram POS features	p_{i-2}, p_{i-1}, p_i	& t_i
	p_{i-1}, p_i, p_{i+1}	& t_i
	p_i, p_{i+1}, p_{i+2}	& t_i
Previous tag	t_{i-1}	& t_i
Tag two back	t_{i-2}	& t_i
Next tag	t_{i+1}	& t_i
Tag two ahead	t_{i+2}	& t_i
Bigram tag features	t_{i-2}, t_{i-1}	& t_i
	t_{i-1}, t_{i+1}	& t_i
	t_{i+1}, t_{i+2}	& t_i

Table 4: Feature templates used in chunking experiments.

(Collins, 2002) and used POS-trigrams as well. Table 4 lists the features used in chunking experiments.

Table 5 shows the results on the development set. Again, bidirectional methods exhibit better performance than unidirectional methods. The difference is bigger with the Start/End representation. Dependency networks did not work well for this chunking task, especially with the Start/End representation.

We applied the best model on the development set in each chunk representation type to the test data. Table 6 summarizes the performance on the test set. Our bidirectional methods achieved F-scores of 93.63 and 93.70, which are better than the best F-score (93.48) of the CoNLL-2000 shared task (Sang and Buchholz, 2000) and comparable to those achieved by other state-of-the-art methods.

5 Discussion

There are some reports that one can improve the performance of unidirectional models by combining outputs of multiple taggers. Shen et al. (2003) reported a 4.9% error reduction of supertagging by

[3] We used the Perl script provided on http://ilk.kub.nl/~ sabine/chunklink/

Representation	Method	Order	Recall	Precision	F-score	Speed (tokens/sec)
IOB2	Left-to-right	1	93.17	93.05	93.11	1,775
		2	93.13	92.90	93.01	989
	Right-to-left	1	92.92	92.82	92.87	1,635
		2	92.92	92.74	92.87	927
	Dependency Networks	1	92.71	92.91	92.81	2,534
		2	92.61	92.95	92.78	1,893
	Easiest-first	1	93.17	93.04	93.11	2,441
		2	93.35	93.32	**93.33**	1,248
	Full Bidirectional	1	93.29	93.14	93.21	712
		2	93.26	93.12	93.19	48
Start/End	Left-to-right	1	92.98	92.69	92.83	861
		2	92.96	92.67	92.81	439
	Right-to-left	1	92.92	92.83	92.87	887
		2	92.89	92.74	92.82	451
	Dependency Networks	1	87.10	89.56	88.32	1,894
		2	87.16	89.44	88.28	331
	Easiest-first	1	93.33	92.95	93.14	1,950
		2	93.31	92.95	93.13	1,016
	Full Bidirectional	1	93.52	93.26	**93.39**	392
		2	93.44	93.20	93.32	4

Table 5: Chunking F-scores on the development set.

Method	Recall	Precision	F-score
SVM (Kudoh and Matsumoto, 2000)	93.51	93.45	93.48
SVM voting (Kudo and Matsumoto, 2001)	93.92	93.89	93.91
Regularized Winnow (with basic features) (Zhang et al., 2002)	93.60	93.54	93.57
Perceptron (Carreras and Marquez, 2003)	93.29	94.19	93.74
Easiest-first (IOB2, second-order)	93.59	93.68	93.63
Full Bidirectional (Start/End, first-order)	93.70	93.65	93.70

Table 6: Chunking F-scores on the test set (Section 20 of the WSJ, 2012 sentences).

pairwise voting between left-to-right and right-to-left taggers. Kudo et al. (2001) attained performance improvement in chunking by conducting weighted voting of multiple SVMs trained with distinct chunk representations. The biggest difference between our approach and such voting methods is that the local classifier in our bidirectional inference methods can have rich information for decision. Also, voting methods generally need many tagging processes to be run on a sentence, which makes it difficult to build a fast tagger.

Our algorithm can be seen as an ensemble classifier by which we choose the highest probability one among the different taggers with all possible decomposition structures. Although choosing the highest probability one is seemingly natural and one of the simplest ways for combining the outputs of different taggers, one could use a different method (e.g. summing the probabilities over the outputs which share the same label sequence). Investigating the methods for combination should be an interesting direction of future work.

As for the computational cost for training, our methods require us to train 2^{2n} types of classifiers when we adopt an nth order markov assumption. In many cases a second-order model is sufficient because further increase of n has little impact on performance. Thus the training typically takes four or 16 times as much time as it would take for training a single unidirectional tagger, which looks somewhat expensive. However, because each type of classifier can be trained independently, the training can be performed completely in parallel and run with the same amount of memory as that for training a single classifier. This advantage contrasts with the case for CRFs which requires substantial amount of memory and computational cost if one tries to incorporate higher-order features about tag sequences.

Tagging speed is another important factor in building a practical tagger for large-scale text mining. Our inference algorithm with the easiest-first strategy needs no Viterbi decoding unlike MEMMs and CRFs, and makes it possible to perform very fast tagging with high precision.

6 Conclusion

We have presented a bidirectional inference algorithm for sequence labeling problems such as POS tagging, named entity recognition and text chunking. The algorithm can enumerate all possible decomposition structures and find the highest probability sequence together with the corresponding decomposition structure in polynomial time. We have also presented an efficient bidirectional inference algorithm based on the easiest-first strategy, which gives comparable performance to full bidirectional inference with significantly lower computational cost.

Experimental results of POS tagging and text chunking show that the proposed bidirectional inference methods consistently outperform unidirectional inference methods and our bidirectional MEMMs give comparable performance to that achieved by state-of-the-art learning algorithms including kernel support vector machines.

A natural extension of this work is to replace the maximum entropy modeling, which was used as the local classifiers, with other machine learning algorithms. Support vector machines with appropriate kernels is a good candidate because they have good generalization performance as a single classifier. Although SVMs do not output probabilities, the easiest-first method would be easily applied by considering the margins output by SVMs as the confidence of local classification.

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden markov support vector machines. In *Proceedings of ICML 2003*, pages 3–10.

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.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference (ANLP)*.

Xavier Carreras and Lluis Marquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. *Technical Report CMUCS -99-108, Carnegie Mellon University*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*, pages 1–8.

Jesus Gimenez and Lluis Marquez. 2003. Fast and accurate part-of-speech tagging: The SVM approach revisited. In *Proceedings of RANLP 2003*, pages 158–165.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP 2003*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL 2001*.

Taku Kudoh and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL-2000*, pages 142–144.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

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.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL 1999*, pages 173–179.

Libin Shen and Aravind K. Joshi. 2003. A SNoW based Supertagger with Application to NP Chunking. In *Proceedings of ACL 2003*, pages 505–512.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Tong Zhang, Fred Damereau, and David Johnson. 2002. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–638.

Context-Based Morphological Disambiguation with Random Fields[*]

Noah A. Smith and **David A. Smith** and **Roy W. Tromble**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{nasmith,dasmith,royt}@cs.jhu.edu

Abstract

Finite-state approaches have been highly successful at describing the morphological processes of many languages. Such approaches have largely focused on modeling the phone- or character-level processes that generate candidate lexical *types*, rather than *tokens* in *context*. For the full analysis of words in context, *disambiguation* is also required (Hakkani-Tür et al., 2000; Hajič et al., 2001). In this paper, we apply a novel source-channel model to the problem of morphological disambiguation (segmentation into morphemes, lemmatization, and POS tagging) for concatenative, templatic, and inflectional languages. The channel model exploits an existing morphological dictionary, constraining each word's analysis to be linguistically valid. The source model is a factored, conditionally-estimated random field (Lafferty et al., 2001) that learns to disambiguate the full sentence by modeling local contexts. Compared with baseline state-of-the-art methods, our method achieves statistically significant error rate reductions on Korean, Arabic, and Czech, for various training set sizes and accuracy measures.

1 Introduction

One of the great successes in computational linguistics has been the construction of morphological analyzers for diverse languages. Such tools take in words and enumerate the possible morphological analyses—typically a sequence of morphemes, perhaps part-of-speech tagged. They are often encoded as finite-state transducers (Kaplan and Kay, 1981; Koskenniemi, 1983; Beesley and Karttunen, 2003).

What such tools do not provide is a means to *disambiguate* a word in *context*. For languages with complex morphological systems (inflective, agglutinative, and polysynthetic languages, for example), a word form may have many analyses. To pick the right one, we must consider the word's context. This problem has been tackled using statistical sequence models for Turkish (Hakkani-Tür et al., 2000) and Czech (Hajič et al., 2001); their approaches (and ours) are not unlike POS tagging, albeit with complex tags.

[*]This work was supported by a Fannie and John Hertz Foundation Fellowship, a NSF Fellowship, and a NDSEG Fellowship (sponsored by ARO and DOD). The views expressed are not necessarily endorsed by sponsors. We thank Eric Goldlust and Markus Dreyer for Dyna language support and Jason Eisner, David Yarowsky, and three anonymous reviewers for comments that improved the paper. We also thank Jan Hajič and Pavel Krbec for sharing their Czech tagger.

In this paper, we describe context-based models for morphological disambiguation that take full account of existing morphological dictionaries by estimating *conditionally* against only dictionary-accepted analyses of a sentence (§2). These models are an instance of conditional random fields (CRFs; Lafferty et al., 2001) and include overlapping features. Our applications include diverse disambiguation frameworks and we make use of linguistically-inspired features, such as local lemma dependencies and inflectional agreement. We apply our model to Korean and Arabic, demonstrating state-of-the-art results in both cases (§3). We then describe how our model can be expanded to complex, structured morphological tagging, including an efficient estimation method, demonstrating performance on Czech (§4).

2 Modeling Framework

Our framework is a source-channel model (Jelinek, 1976). The *source* (modeled probabilistically by p_s) generates a sequence of unambiguous tagged morphemes $\mathbf{y} = \langle y_1, y_2, ...\rangle \in \mathcal{Y}^+$ ($\mathcal{Y}$ is the set of unambiguous tagged morphemes in the language).[1] The precise contents of the tag will vary by language and corpus but will minimally include POS. $\mathbf{y}$ passes through a *channel* (modeled by p_c), which outputs $\mathbf{x} = \langle x_1, x_2, ...\rangle \in (\mathcal{X} \cup \{\textsc{oov}\})^+$, a sequence of surface-level words in the language and out-of-vocabulary words (OOV; $\mathcal{X}$ is the language's vocabulary). Note that $|\mathbf{x}|$ may be smaller than $|\mathbf{y}|$, since some morphemes may combine to make a word. We will denote by $\mathbf{y}_i$ the contiguous subsequence of $\mathbf{y}$ that generates x_i; $\vec{y}$ will refer to a dictionary-recognized *type* in $\mathcal{Y}^+$.

At test time, we *decode* the observed $\mathbf{x}$ into the most probable sequence of tag/morpheme pairs:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} p_s(\mathbf{y}) \cdot p_c(\mathbf{x} \mid \mathbf{y}) \quad (1)$$

Training involves constructing p_s and p_c. We assume that there exists a training corpus of text (each word x_i annotated with its correct analysis $\mathbf{y}_i^*$) and a morphological dictionary. We next describe the channel model and the source model.

[1]The sequence also includes segmentation markings between words, not shown to preserve clarity.

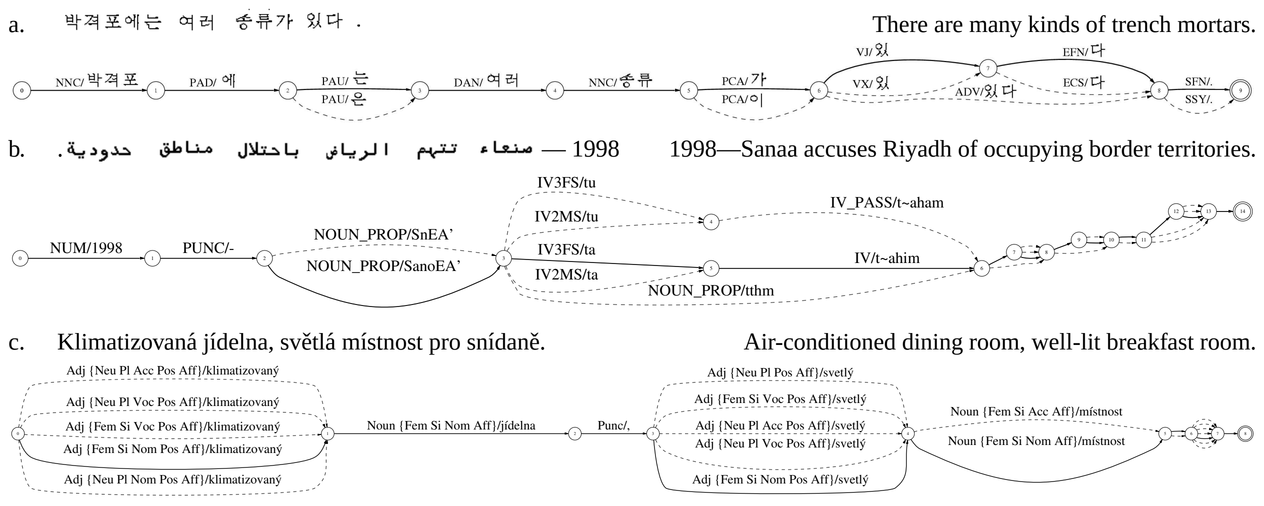

a. 박격포에는 여러 종류가 있다 .
There are many kinds of trench mortars.
NNC/박격포
PAD/에
PAU/는
PAU/은
DAN/여러
NNC/종류
PCA/가
PCA/이
VJ/있
VX/있
ADV/있다
EFN/다
ECS/다
SFN/.
SSY/.
b. .حدودية مناطق باحتلال الرياض تتهم صنعاء — 1998
1998—Sanaa accuses Riyadh of occupying border territories.
NUM/1998
PUNC/-
NOUN_PROP/SnEA'
NOUN_PROP/SanoEA'
IV3FS/tu
IV2MS/tu
IV3FS/ta
IV2MS/ta
NOUN_PROP/tthm
IV_PASS/t~aham
IV/t~ahim
c. Klimatizovaná jídelna, světlá místnost pro snídaně.
Air-conditioned dining room, well-lit breakfast room.
Adj {Neu Pl Acc Pos Aff}/klimatizovaný
Adj {Neu Pl Voc Pos Aff}/klimatizovaný
Adj {Fem Si Voc Pos Aff}/klimatizovaný
Adj {Fem Si Nom Pos Aff}/klimatizovaný
Adj {Neu Pl Nom Pos Aff}/klimatizovaný
Noun {Fem Si Nom Aff}/jídelna
Punc/,
Adj {Neu Pl Pos Aff}/svetlý
Adj {Fem Si Voc Pos Aff}/svetlý
Adj {Neu Pl Acc Pos Aff}/svetlý
Adj {Neu Pl Voc Pos Aff}/svetlý
Adj {Fem Si Nom Pos Aff}/svetlý
Noun {Fem Si Acc Aff}/místnost
Noun {Fem Si Nom Aff}/místnost

Figure 1: Lattices for example sentences in Korean (a), Arabic (b), and Czech (c). Arabic lemmas are not shown, and some Arabic and Czech arcs are unlabeled, for readability. The Arabic morphemes are shown in Buckwalter's encoding. The arcs in the correct path through each lattice are solid (incorrect arcs are dashed). Note the adjective-noun agreement in the correct path through the Czech lattice (c). The Czech lattice has no lemma-ambiguity; this is typical in Czech (see §4).

2.1 Morphological dictionaries and the channel

A great deal of research has gone into developing morphological analysis tools that enumerate valid analyses $\vec{y} \in \mathcal{Y}^+$ for a particular word $x \in \mathcal{X}$. Typically these tools are unweighted and therefore do not enable token disambiguation.[2]

They are available for many languages. We will refer to this source of categorial lexical information as a morphological dictionary d that maps $\mathcal{X} \to 2^{\mathcal{Y}^+}$. The set $d(x)$ is the set of analyses for word x; the set $d(\mathbf{x})$ is the set of whole-sentence analyses for sentence $\mathbf{x} = \langle x_1, x_2, ...\rangle$.

$d(\mathbf{x})$ can be represented as an acyclic lattice with a "sausage" shape familiar from work in speech recognition (Mangu et al., 1999). Note that for languages with bound morphemes, $d(x)$ will consist of a set of sequences of tokens, so a given "link" in the sausage lattice may contain paths of different lengths. Fig. 1 shows sausage lattices for sentences in three languages.

In this paper, the dictionary defines the support set of the channel model. That is, $p_c(\mathbf{x} \mid \mathbf{y}) > 0$ if and only if $\mathbf{y} \in d(\mathbf{x})$. This is a clean way to incorporate domain knowledge into the probabilistic model; this kind of constraint has been applied in previous work at decoding time (Hakkani-Tür et al., 2000; Hajič et al., 2001). In such a model, each word is independent of its neighbors (because the dictionary ignores context).

Estimation. A *unigram* channel model defines

$$p_c(\mathbf{x} \mid \mathbf{y}) \overset{\text{def}}{=} \prod_{i=1}^{|\mathbf{x}|} p(x_i \mid \mathbf{y}_i) \qquad (2)$$

The simplest estimate of this model is to make $p(\cdot, \cdot)$ *uniform* over $(x, \vec{y})$ such that $\vec{y} \in d(x)$. Doing so and marginalizing to get $p(x \mid \vec{y})$ makes the channel model encode categorial information only, leaving all learning to the source model.[3]

Another way to estimate this model is, of course, from data. This is troublesome, because—modulo optionality—x is expected to be *known* given $\vec{y}$, resulting in a huge model with mostly 1-valued probabilities. Our solution is to take a *projection* π of $\vec{y}$ and let $p(\cdot \mid \vec{y}) \approx p(\cdot \mid \pi(\vec{y}))$. In this paper, π maps the analysis to its morphological tag (or tag sequence). We will refer to this as the "tag channel."

OOV. Morphological dictionaries typically do not have complete coverage of a language. We can augment them in two ways using the training data. If a known word x (one for which $d(x)$ is non-empty) appears in the training dataset with an analysis not in $d(x)$, we add the entry to the dictionary. Unknown words (those not recognized by the dictionary) are replaced by an OOV symbol. $d(\text{OOV})$ is taken to be the set of all analyses for any OOV word seen in training. Rather than attempt to recover the morpheme sequence for an OOV word, in this paper we try only for the tag sequence, replacing all of an OOV's morphemes with the OOV symbol. Since OOV symbols account for less than 2% of words in our corpora, we leave

[2] *Probabilistic* modeling of what we call the morphological channel was first carried out by Levinger et al. (1995), who used unlabeled data to estimate $p(\vec{y} \mid x)$ for Hebrew, with the support defined by a dictionary.

[3] Note that this makes the channel term in Eq. 1 a constant. Then decoding means maximizing $p_s(\mathbf{y})$ over $\mathbf{y} \in d(\mathbf{x})$, equivalently maximizing $p(\mathbf{y} \mid d(\mathbf{x}))$.

more sophisticated channel models to future work.

2.2 The source model

The source model p_s defines a probability distribution over $\mathcal{Y}^+$, sequences of (tag, morpheme) pairs. Our source models can be viewed as weighted multi-tape finite-state automata, where the weights are associated with local, often overlapping features of the path through the automaton.

Estimation. We estimate the source *conditionally* from annotated data. That is, we maximize

$$\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}^+\times\mathcal{Y}^+} \tilde{p}(\mathbf{x},\mathbf{y}) \log p_s\left(\mathbf{y} \mid d(\mathbf{x}), \vec{\theta}\right) \quad (3)$$

where $\tilde{p}(\cdot,\cdot)$ is the empirical distribution defined by the training data and $\vec{\theta}$ are the model parameters. In terms of Fig. 1, our learner maximizes the weight of the correct (solid) path through each lattice, at the expense of the other incorrect (dashed) paths. Note that

$$\log p_s\left(\mathbf{y} \mid d(\mathbf{x}), \vec{\theta}\right) = \log \frac{p_s\left(\mathbf{y} \mid \vec{\theta}\right)}{\sum_{\mathbf{y}'\in d(\mathbf{x})} p_s\left(\mathbf{y}' \mid \vec{\theta}\right)} \quad (4)$$

The sum in the denominator is computed using a dynamic programming algorithm (akin to the forward algorithm); it involves computing the sum of all paths through the "sausage" lattice of possible analyses for $\mathbf{x}$. By doing this, we allow knowledge of the support of the *channel* model to enter into our estimation of the *source* model. It is important to note that the *estimation* of the model (the objective function used in training, Eq. 3) is distinct from the source-channel *structure* of the model (Eq. 1).

The lattice-conditional estimation approach was first used by Kudo et al. (2004) for Japanese segmentation and hierarchical POS-tagging and by Smith and Smith (2004) for Korean morphological disambiguation. The resulting model is an instance of a *conditional random field* (CRF; Lafferty et al., 2001). When training a CRF for POS tagging, IOB chunking (Sha and Pereira, 2003), or word segmentation (Peng et al., 2004), one typically structures the conditional probabilities (in the objective function) using domain knowledge: in POS tagging, the set of allowed tags for a word is used; in IOB chunking, the bigram "O I" is disallowed; and in segmentation, a lexicon is used to enumerate the possible word boundaries.[4]

[4]This refinement is in the same vein as the move from *maximum likelihood* estimation to *conditional* estimation. MLE would make the sum in the denominator of Eq. 4 $\mathcal{Y}^+$, which for log-linear models is often intractable to compute (and for sequence models may not converge). Conditional estimation limits the sum to the subset of $\mathcal{Y}^+$ that is consistent with $\mathbf{x}$, and our variant further stipulates consistency with the dictionary entries for $\mathbf{x}$.

Our approach is the same, with two modifications. First, we model the relationship between labels y_i and words x_i in a separately-estimated channel model (§2.1). Second, our labels are complex. Each word x_i is tagged with a *sequence* of one or more tagged morphemes; the tags may include multiple fields. This leads to models with more parameters. It also makes the dictionary especially important for limiting the size of the sum in the denominator, since a complex label set $\mathcal{Y}$ could in principle lead to a huge hypothesis space for a given sentence $\mathbf{x}$. Importantly, it makes training conditions more closely match testing conditions, ruling out hypotheses a dictionary-aware decoder would never consider.

Optimization. The objective function (Eq. 3) is concave and known to have a unique global maximum. Because log-linear models and CRFs have been widely described elsewhere (e.g., Lafferty, 2001), we note only that we apply a standard first-order numerical optimization method (L-BFGS; Liu and Nocedal, 1989). The structure, features, and regularization of our models will be described in §3 and §4.

Prior work (morphological source models). Hakkani-Tür et al. (2000) described a system for Turkish that was essentially a source model; Hajič et al. (2001) described an HMM-based system for Czech that could be viewed as a combined source and channel. Both used dictionaries and estimated their (generative) models using maximum likelihood (with smoothing).[5] Given enough data, a ML-estimated model will learn to recognize a good path $\mathbf{y}$, but it may not learn to discriminate a good $\mathbf{y}$ from wrong alternatives *per se*. The generative framework is limiting as well, disallowing the straightforward inclusion of arbitrary overlapping features. We present a competitive Czech model in §4.

3 Concatenative Models

The beauty of log-linear models is that estimation is straightforward given *any* features, even ones that are not orthogonal (i.e., "overlap"). This permits focusing on feature (or feature template) selection without worries about the mathematics of training.

We consider two languages modeled by concatenative processes with surface changes at morpheme boundaries: Korean and Arabic.

Our model includes features for tag n-grams, morpheme n-grams, and pairs of the two (possibly of different lengths and offsets). Fig. 2 illustrates TM3, our base model. TM3 includes feature templates for some tuples of three or fewer elements, plus begin and end templates.

[5]Hajič et al. also included a rule-based system for pruning hypotheses, which gave slight performance gains.

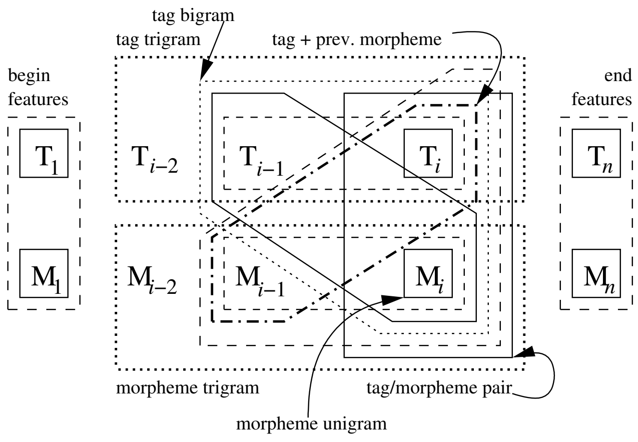


Figure 2: The base two-level trigram source model, TM3. Each polygon corresponds to a feature template. This is a two level, second-order Markov model (weighted finite-state machine) parameterized with overlapping features. Note that only some features are labeled in the diagram.

A variant, TM3H, includes all of the same templates, plus a similar set of templates that look only at *head* morphemes. For instance, a feature fires for each trigram of heads, even though there are (bound) morphemes between them. This increases the domain of locality for semantic content-bearing morphemes. This model requires slight changes to the dynamic programming algorithms for inference and training (the previous two heads must be remembered at each state).

Every instantiation of the templates seen in *any* lattice $d(\mathbf{x})$ built from training data is included in the model, not just those seen in correct analyses $\mathbf{y}^*$.[6]

3.1 Experimental design

In all of our experiments, we vary the training set size and the amount of smoothing, which is enforced by a diagonal Gaussian prior (L_2 regularizer) with variance σ^2. The $\sigma^2 = \infty$ case is equivalent to not smoothing. We compare performance to the expected performance of a randomized baseline that picks for each word token x an analysis from $d(x)$; this gives a measure of the amount of ambiguity and is denoted "channel only." Performance of unigram, bigram, and trigram HMMs estimated using maximum likelihood (barely smoothed, using add-10^{-14}) is also reported. (The unigram HMM simply picks the most likely $\vec{y}$ for each x, based on training data and is so marked.)

In the experiments in this section, we report three performance measures. *Tagging* accuracy is the fraction of words whose tag sequence was correctly identified in entirety; *morpheme* accuracy is defined analogously. *Lemma* accuracy is the fraction of words whose lemma was correctly identified.

3.2 Korean experiments

We applied TM3 and TM3H to Korean. The dataset is the Korean Treebank (Han et al., 2002), with up to 90% used for training and 10% (5K words) for test. The morphological dictionary is `klex` (Han, 2004). There are 27 POS tags in the tag set; the corpus contains 10K word types and 3,272 morpheme types. There are 1.7 morphemes per word token on average ($\sigma = 0.75$). A Korean word generally consists of a head morpheme with a series of enclitic suffixes. In training the head-augmented model TM3H, we assume the first morpheme of every word is the head and lemma.

Results are shown in Tab. 1. TM3H achieved very slight gains over TM3, and the tag channel model was helpful only with the smaller training set. The oracle (last line of Tab. 1) demonstrates that the coverage of the dictionary remains an obstacle, particularly for recovering morphemes. Another limitation is the small amount of training data, which may be masking differences among estimation conditions. We report the performance of TM3H with "factored" estimation. This will be discussed in detail in §4; it means that a model containing *only* the head features was trained on its own, then combined with the independently trained TM3 model at test time. Factored training was slightly faster and did not affect performance at all; accuracy scores were identical with unfactored training.

Prior work (Korean). Similar results were presented by Smith and Smith (2004), using a similar estimation strategy with a model that included far more feature templates. TM3 has about a third as many parameters and TM3H about half; performance is roughly the same (numbers omitted for space). Korean disambiguation results were also reported by Cha et al. (1998), who applied a deterministic morpheme pattern dictionary to segment words, then used a bigram HMM tagger. They also applied transformation-based learning to fix common errors. Due to differences in tag set and data, we cannot compare to that model; a bigram baseline is included.

3.3 Arabic experiments

We applied TM3 and TM3H to Arabic. The dataset is the Arabic Treebank (Maamouri et al., 2003), with up to 90% used for training and 10% (13K words) for test. The morphological dictionary is Buckwalter's analyzer (version 2), made available by the LDC (Buckwalter, 2004).[7] This analyzer has total coverage of the corpus; there are no

[6]If we used only features observed to occur in $\mathbf{y}^*$, we would not be able to learn negative weights for *unlikely* bits of structure seen in the lattice $d(\mathbf{x})$ but not in $\mathbf{y}^*$.

[7]Arabic morphological processing was also addressed by Kiraz (2000), who gives a detailed review of symbolic work in that area, and by Darwish (2002).

			Korean						Arabic								
			POS tagging accuracy		morpheme accuracy		lemma accuracy		POS tagging accuracy			morpheme accuracy			lemma accuracy		
		σ^2	32K	49K	32K	49K	32K	49K	38K	76K	114K	38K	76K	114K	38K	76K	114K
	most likely $\vec{y}$		86.0	86.9	87.5	88.8	95.3	95.7	84.5	87.0	88.3	83.2	86.2	87.0	37.9	39.8	40.9
uniform channel	channel only		62.6	62.6	70.3	70.8	86.4	86.4	43.7	43.7	43.7	41.2	41.2	41.2	27.2	27.2	27.2
	bigram HMM		90.7	91.2	83.2	86.1	96.9	97.2	90.3	92.0	92.8	89.2	91.4	91.6	85.7*	87.8*	87.9*
	trigram HMM		91.5	91.8	83.3	86.0	97.0	97.2	89.8	92.0	93.0	88.5	91.3	91.3	85.2*	87.8*	87.7*
	TM3	∞	90.7	91.3	**89.3**	**90.5**	97.1	97.4	**94.6**	**95.4**	**95.9**	**93.4**	**94.3**	**94.9**	**89.7***	**90.5***	**90.7***
		10	91.2	91.7	**89.4**	**90.6**	97.1	**97.6**	**95.3**	**95.7**	**96.1**	**93.9**	**94.5**	**95.0**	**90.2***	**90.6***	**91.1***
		1	91.5	92.2	**89.4**	**90.6**	97.1	**97.5**	**95.2**	**95.7**	**96.0**	**93.9**	**94.5**	**94.7**	**90.0***	**90.7***	**91.0***
	TM3H	∞	91.1	91.1	**89.3**	**90.4**	97.2	97.5	**95.0**	**95.7**	**96.0**	**94.0**	**94.8**	**95.3**	**93.3**	**93.9**	**94.2**
	(factored)	10	91.3	91.9	**89.5**	**90.6**	**97.3**	**97.6**	**95.3**	**95.7**	**96.1**	**94.2**	**94.7**	**95.4**	**93.4**	**93.6**	**94.4**
		1	91.4	92.2	**89.5**	**90.7**	**97.3**	**97.6**	**95.4**	**95.8**	**96.1**	**94.4**	**94.8**	**95.1**	**93.3**	**93.8**	**94.2**
tag channel	channel only		51.4	51.3	60.6	60.4	81.2	81.7	41.4	40.6	40.1	39.9	39.1	38.6	26.7*	26.5*	26.4*
	bigram HMM		91.2	90.9	**88.9**	**90.1**	97.0	97.3	**91.0**	**92.3**	**93.4**	**89.7**	91.5	**91.9**	**88.1***	**89.9***	**90.0***
	trigram HMM		91.6	91.9	**88.9**	**90.2**	97.1	**97.4**	**91.1**	**92.9**	**93.7**	**89.6**	**92.2**	**92.0**	**88.1***	**90.6***	**90.4***
	TM3	∞	90.8	91.0	**89.5**	**90.5**	**97.4**	97.5	**95.1**	**95.7**	**96.0**	**93.8**	**94.6**	**95.0**	**92.2***	**93.1***	**93.2***
		10	90.6	91.1	**89.5**	**90.7**	97.2	**97.6**	**95.2**	**95.6**	**96.0**	**93.9**	**94.7**	**95.0**	**92.4***	**93.2***	**93.5***
		1	90.1	90.9	**89.5**	**90.7**	97.1	**97.6**	**94.9**	**95.5**	**95.8**	**93.8**	**94.5**	**94.8**	**92.2***	**93.0***	**93.1***
	TM3H	∞	91.0	91.0	**89.4**	**90.5**	97.2	**97.6**	**95.1**	**95.8**	**96.0**	**94.0**	**95.1**	**95.4**	**93.3**	**94.3**	**94.4**
	(factored)	10	90.4	91.2	**89.6**	**90.7**	**97.4**	**97.6**	**95.2**	**95.7**	**96.0**	**94.1**	**94.8**	**95.4**	**93.3**	**94.0**	**94.6**
		1	90.1	91.0	**89.5**	**90.7**	**97.3**	**97.6**	**95.1**	**95.5**	**95.9**	**94.1**	**94.9**	**95.1**	**93.3**	**94.0**	**94.4**
	oracle given $d(\mathbf{x})$		95.3	95.7	90.2	91.2	98.1	98.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 1: Korean (left, 5K test-set) and Arabic (right, 13K test-set) disambiguation. A word is marked correct only if its entire tag (or morpheme) sequence (or lemma) was correctly identified. Morpheme and lemma accuracy do not include OOV words. The oracle is an upper bound on accuracy given the morphological dictionary. *These models do not explicitly predict lemmas; the lemma is chosen arbitrarily from those that match the hypothesized tag/morpheme sequence for each word. **Bold** scores indicate a significant improvement over the trigram HMM (binomial sign test, $p < 0.05$).

OOV words. There are 139 distinct POS tags; these contain some inflectional information which we treat atomically. For speed, TM3H was trained in two separate pieces: TM3 and the lemma features added by TM3H.

Arabic has a templatic morphology in which consonantal roots are transformed into surface words by the insertion of vowels and ancillary consonants. Our system does not model this process except through the use of Buckwalter's dictionary to define the set of analyses for each word (cf., Daya et al., 2004, who modeled interdigitation in Hebrew). We treat the analysis of an Arabic word as a sequence $\vec{y}$ of pairs of morphemes and POS tags, plus a lemma. The lemma, given in the dictionary, provides further disambiguation beyond the head morpheme. The lemma is a standalone dictionary headword and not merely the consonantal root, as in some other work. The "heads" modeled by TM3H correspond to these lemmas. There are 20K word types, and 34K morpheme types. There are 1.7 morphemes per word token on average ($\sigma = 0.77$).

Results are shown in Tab. 1. Across tasks and training set sizes, our models reduce error rates by more than 36% compared to the trigram HMM source with tag channel. The TM3H model and the tag channel offer slight gains over the base TM3 model (especially on lemmatization), though the tag channel offers no help in POS tagging.

Prior work (Arabic). Both Diab et al. (2004) and Habash and Rambow (2005) use support-vector machines with local features; the former for tokenization, POS tagging, and base phrase chunking; the latter for full morphological disambiguation. Diab et al. report results for a coarsened 24-tag set, while we use the full 139 tags from the Arabic Treebank, so the systems are not directly comparable. Habash and Rambow present even better results on the same POS tag set. Our full disambiguation results appear to be competitive with theirs. Khoja (2001) and Freeman (2001) describe Arabic POS taggers and many of the issues involved in developing them, but because tagged corpora did not yet exist, there are no comparable quantitative results.

4 Czech: Model and Experiments

Inflective languages like Czech present a new set of challenges. Our treatment of Czech is not concatenative; following prior work, the analysis for each word x is a single tag/lemma pair y. Inflectional affixes in the surface form are represented as features in the tag. While lemmatization of Czech is not hard (there is little ambiguity), tagging is quite difficult, because morphological tags are highly complex. Our tag set is the Prague Dependency Treebank (PDT; Hajič, 1998) set, which consists of fifteen-field tags that indicate POS as well as inflectional information (case, number, gender, etc.). There are over

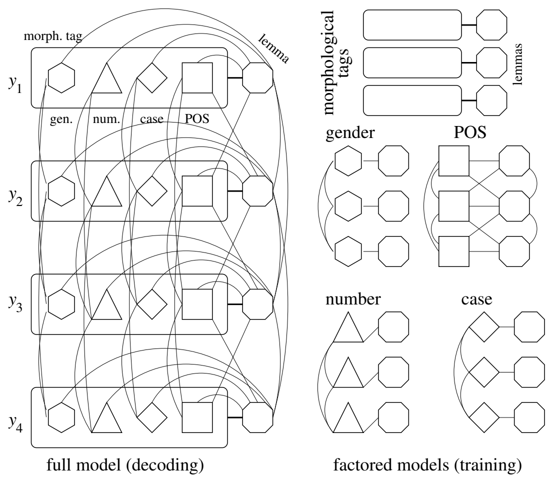


Figure 3: The Czech model, shown as an undirected graphical model. The structure of the full model is on the left; factored components for estimation are shown on the right. Each of these five models contains a subset of the TM3 features. The full model is only used to decode. The factored models make training faster and are used for pruning.

1,400 distinct tag types in the PDT.

Czech has been treated probabilistically before, perhaps most successfully by Hajič et al. (2001).[8] In contrast, we estimate conditionally (rather than by maximum likelihood for a generative HMM) and separate the training of the source and the channel. We also introduce a novel *factored* treatment of the morphological tags.

4.1 Factored tags and estimation

Because Czech morphological tags are not monolithic, the choice among them can be treated as several more or less orthogonal decisions. The case feature of one word, for example, is expected to be conditionally independent of the next word's gender, given the next word's case. Constraints in the language are expected to cause features like case, number, and gender to agree locally (on words that have such features) and somewhat independently of each other. Coarser POS tagging may be treated as another, roughly independent stream.

Log-linear models and the use of a morphological dictionary make this kind of tag factoring possible. Our approach is to separately train five log-linear models. Each model is itself an instance of some of the templates from TM3, modeling a projection of the full analysis. The model and its factored components are illustrated in Fig. 3.

POS model. The full tag is replaced by the POS tag (the first two fields); there are 60 POS tags. The TM3 feature templates are included twice: once for the full tag and once for a coarser tag (the first PDT field, for which there are 12 possible values).[9]

Gender, number, and case models. The full tag is replaced by the gender (or case or number) field. This model includes bigrams and trigrams as well as field-morpheme unigram features. These models are intended to learn to predict local agreement.

Tag-lemma model. This model contains unigram features of full PDT tags, both alone and with lemmas. It is intended to learn to penalize morphological tags that are rare, or that are rare with a particular lemma. In our formulation, this is *not* a channel model, because it ignores the surface word forms.

Each model is estimated independently of the others. The lattice $d(\mathbf{x})$ against which the conditional probabilities are estimated contains the relevant *projection* of the full morphological tags (with lemmas). To decode, we run a Viterbi-like algorithm that uses the union of all models' features to pick the best analysis (full morphological tags and lemmas) allowed by the dictionary.

There are two important advantages of factored training. First, each model is faster to train alone than a model with all features merged; in fact, training the fully merged model takes far too long to be practical. Second, factored models can be held out at test time to measure their effect on the system, without retraining.

Prior work (factored training). Separately training different models that predict the same variables (e.g., $\mathbf{x}$ and $\mathbf{y}$) then combining them for consensus-based inference (either through a mixture or a product of probabilities) is an old idea (Genest and Zidek, 1986). Recent work in learning weights for the component "expert" models has turned to *cooperative* techniques (Hinton, 1999). Decoding that finds $\mathbf{y}$ (given $\mathbf{x}$) to maximize some weighted average of log-probabilities is known as a *logarithmic opinion pool* (LOP). LOPs were applied to CRFs (for named entity recognition and tagging) by Smith et al. (2005), with an eye toward regularization. Their experts (each a CRF) contained overlapping feature sets, and the combined model achieved much the same effect as training a single model with smoothing. Note that our models, unlike theirs, *partition* the feature space; there is only one CRF, but some parameters are ignored when estimating other parameters. We have not estimated log-domain mixing coefficients—we weight all models' contributions equally. Sutton and McCallum (2005) have applied factored estimation to CRFs, motivated (like us) by speed; they also describe how factored estimation

[8]Czech morphological processing was studied by Petkevič (2001), Hlaváčová (2001) (who focuses on handling OOV words), and Mráková and Sedlacek (2003) (who use partial parsing to reduce the set of possible analyses), *inter alia*.

[9]Lemma-trigram and fine POS-unigram/lemma-bigram features were eliminated to limit model size.

	σ^2	full morph. accuracy 376K	full morph. accuracy 768K	lemma accuracy 376K	lemma accuracy 768K	POS accuracy 376K	POS accuracy 768K	OOV POS accuracy 376K	OOV POS accuracy 768K
channel only		61.4	60.3	85.1	84.2	88.5	87.2	17.8	16.4
most likely $\vec{y}$		80.0	80.8	98.1	98.1	97.9	97.8	52.0	52.0
Hajič et al. HMM		**88.8**	**89.2**	97.9	97.9	95.8	95.8	52.0	52.0
+ OOV model		90.5	90.8	97.9	97.9	96.7	96.6	93.0	92.9
full	∞	88.1	88.5	**98.3**	**98.5**	**98.3**	**98.3**	**60.2**	**61.8**
oracle given pruning		98.6	99.3	99.5	99.6	99.1	99.7	60.2	90.3
	10	88.4	88.5	**98.4**	**98.4**	**98.3**	**98.2**	**61.8**	**59.4**
oracle given pruning		99.3	99.3	99.5	99.6	99.8	99.7	93.4	90.6
	1	88.6	88.6	**98.4**	**98.4**	**98.2**	**98.1**	**60.0**	**56.7**
oracle given pruning		99.3	99.3	99.5	99.6	99.8	99.8	95.0	94.0
– POS	∞	87.9	88.0†	**98.2**	**98.2**†	**98.0**	**97.9**†	**55.7**	51.7†
	10	88.1	88.3†	**98.2**	**98.3**†	**98.0**	**97.9**†	**55.4**	51.6†
	1	88.4	88.5†	**98.2**	**98.2**†	**98.0**	**97.9**†	**55.0**	51.9†
– tag-lemma	∞	87.8	88.3	**98.3**	**98.6**	**98.3**	**98.3**	**60.2**	**59.7**
	10	88.0	88.1	**98.4**	**98.5**	**98.3**	**98.2**	**59.1**	**59.1**
	1	88.0	88.1	**98.4**	**98.4**	**98.2**	**98.1**	**59.0**	**58.1**
POS only	∞	65.6*	65.5*	**98.3**	**98.6**	**98.3**	**98.4**	**60.2**	**63.7**
	10	65.7*	65.5*	**98.5**	**98.6**	**98.5**	**98.5**	**65.2**	**66.4**
	1	65.7*	65.5*	**98.6**	**98.7**	**98.6**	**98.6**	**67.2**	**67.2**
POS & tag-lemma†	∞	81.2	82.3	**98.3**	**98.6**	**98.3**	**98.4**	**60.2**	**63.9**
	10	81.9	82.3	**98.5**	**98.6**	**98.4**	**98.5**	**65.8**	**67.2**
	1	82.0	82.3	**98.4**	**98.5**	**98.5**	**98.4**	**67.8**	**66.3**
oracle given $d(\mathbf{x})$		99.8	99.8	99.5	99.6	99.9	99.9	100.0	100.0

Table 2: Czech disambiguation: test-set (109K words) accuracy. A word is marked correct only if its entire morphological tag (or morpheme or POS tag) was correctly identified. Note that the full tag is a complex, 15-field morphological label, while "POS" is a projection down to a tagset of size 60. Lemma accuracy does not include OOV words. *The POS-only model selects only POS, not full tags; these measures are expected performance if the full tag is selected randomly from those in the dictionary that match the selected POS. †Required more aggressive pruning. **Bold** scores were significantly better than the HMM of Hajič et al. (binomial sign test, $p < 0.05$). Our models were slightly but significantly worse on full tagging, but showed significant improvements on recovering POS tags and lemmas.

maximizes a lower bound on the unfactored objective. Smith and Smith (2004) applied factored estimation to a bilingual weighted grammar, driven by data limitations.

4.2 Experiments

Our corpus is the PDT (Hajič, 1998), with up to 60% used for training and 10% (109K words) used for test.[10] The morphological dictionary is the one packaged with the PDT; it covers about 98% of the tokens in the corpus. The remaining 2% have (unsurprisingly) a diverse set of 300–400 distinct tags, depending on the training set size.[11]

Results are shown in Tab. 2. We compare to the HMM of (Hajič et al., 2001) *without* its OOV component.[12] We report morphological tagging accuracy on words; we also report lemma accuracy (on non-OOV words), POS accuracy on all words, and POS accuracy on OOV words. The channel model (not shown) tended to have a small, harmful effect on performance.

Without any explicit OOV treatment, our POS-only component model significantly reduces lemma and POS errors compared to Hajič et al.'s model. On recovering *full* morphological tags, our *full* model is close in performance to Hajič et al., but still significantly worse. It is likely that for many tasks, these performance gains are more helpful than the loss on full tagging is harmful.

Why doesn't our full model perform as well as Hajič et al.'s model? An error analysis reveals that our full model (768K, $\sigma^2 = 1$), compared to the HMM (768K) had 91% as many number errors but 0.1% more gender and 31% more case errors. Taking out those three models ("POS & tag-lemma" in Fig. 2) is helpful on all measures except full tagging accuracy, due in part to substantially increased errors on gender (87% increase), case (54%), and number (35%). The net effect of these components, then, is helpful, but not quite helpful enough to match a well-smoothed HMM on complex tagging. We compared the models on the training set and found the same pattern, demonstrating that this is not merely a matter of over-fitting.

5 Future Work

Two clear ways to improve our models present themselves. The first is better OOV handling, perhaps through an improved channel model. Possibilities include learning weights to go inside the FST-encoded dictionaries and

[10]We used less than the full corpus to keep training time down; note that the training sets are nonetheless substantially larger than in the Korean and Arabic experiments.

[11]During training, these project down to manageable numbers of hypotheses in the factored models. At test-time, however, Viterbi search is quite difficult when OOV symbols occur consecutively. To handle this, we prune OOV arcs from the lattices using the factored POS and inflectional models. For each OOV, every model prunes a projection of the analysis (e.g., the POS model prunes POS tags) until 90% of the posterior mass or 3 arcs remain (whichever is more conservative). Viterbi decoding is run on a lattice containing OOV arcs consistent with the pruned projected lattices.

[12]Results *with* the OOV component are also reported in Tab. 2, but we cannot guarantee their experimental validity, since the OOV component is pre-trained and may have been trained on data in our test set.

directly modeling spelling changes. The second is to turn our factored model into a LOP. Training the mixture coefficients should be straightforward (if time-consuming) with a development dataset.

A drawback of our system (especially for Czech) is that some components (most notably, the Czech POS model) take a great deal of time to train (up to two weeks on 2GHz Pentium systems). Speed improvements are expected to come from eliminating some of the overlapping feature templates, generalized speedups for log-linear training, and perhaps further factoring.

6 Conclusion

We have explored morphological disambiguation of diverse languages using log-linear sequence models. Our approach reduces error rates significantly on POS tagging (Arabic and Czech), morpheme sequence recovery (Korean and Arabic), and lemmatization (all three languages), compared to baseline state-of-the-art methods For complex analysis tasks (e.g., Czech tagging), we have demonstrated that factoring a large model into smaller components can simplify training and achieve excellent results. We conclude that a *conditionally*-estimated source model informed by an existing morphological dictionary (serving as an unweighted channel) is an effective approach to morphological disambiguation.

References

K. R. Beesley and L. Karttunen. 2003. *Finite State Morphology*. CSLI.

T. Buckwalter. 2004. Arabic morphological analyzer version 2.0. LDC2004L02.

J. Cha, G. Lee, and J.-H. Lee. 1998. Generalized unknown morpheme guessing for hybrid POS tagging of Korean. In *Proc. of VLC*.

K. Darwish. 2002. Building a shallow Arabic morphological analyser in one day. In *Proc. of ACL Workshop on Computational Approaches to Semitic Languages*.

E. Daya, D. Roth, and S. Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proc. of EMNLP*.

M. Diab, K. Hacioglu, and D. Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proc. of HLT-NAACL*.

A. Freeman. 2001. Brill's POS tagger and a morphology parser for Arabic. In *Proc. of ACL Workshop on Arabic Language Processing*.

C. Genest and J. V. Zidek. 1986. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1:114–48.

N. Habash and O. Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proc. of ACL*.

J. Hajič, P. Krbec, P. Květoň, K. Oliva, and V. Petkevič. 2001. Serial combination of rules and statistics: A case study in Czech tagging. In *Proc. of ACL*.

J. Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*.

D. Z. Hakkani-Tür, K. Oflazer, and G. Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proc. of COLING*.

C.-H. Han, N.-R. Han, E.-S. Ko, H. Yi, and M. Palmer. 2002. Penn Korean Treebank: Development and evaluation. In *Proc. Pacific Asian Conf. Language and Comp.*

N.-R. Han. 2004. Klex: Finite-state lexical transducer for Korean. LDC2004L01.

G. Hinton. 1999. Products of experts. In *Proc. of ICANN*.

J. Hlavácová. 2001. Morphological guesser of Czech words. In *Proc. of TSD*.

F. Jelinek. 1976. Continuous speech recognition by statistical methods. *Proc. of the IEEE*, 64(4):532–557.

R. M. Kaplan and M. Kay. 1981. Phonological rules and finite-state transducers. Presented at Linguistic Society of America.

S. Khoja. 2001. APT: Arabic part-of-speech tagger. In *Proc. of NAACL Student Workshop*.

G. Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: A case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105.

K. Koskenniemi. 1983. Two-level morphology: A general computational model of word-form recognition and production. Technical Report 11, University of Helsinki.

T. Kudo, K. Yamamoto, and Y. Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

M. Levinger, U. Ornan, and A. Itai. 1995. Learning morpholexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21(3):383–404.

D. C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–28.

M. Maamouri, A. Bies, H. Jin, and T. Buckwalter. 2003. Arabic Treebank part 1 version 2.0. LDC2003T06.

L. Mangu, E. Brill, and A. Stolcke. 1999. Finding consensus among words: Lattice-based word error minimization. In *Proc. of ECSCT*.

E. Mráková and R. Sedlacek. 2003. From Czech morphology through partial parsing to disambiguation. In *Proc. of CLITP*.

F. Peng, F. Feng, and A. McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proc. of COLING*.

V. Petkevič. 2001. Grammatical agreement and automatic morphological disambiguation of inflectional languages. In *Proc. of TSD*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

D. A. Smith and N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*.

A. Smith, T. Cohn, and M. Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proc. of ACL*.

C. Sutton and A. McCallum. 2005. Cliquewise training for undirected models. In *Proc. of UAI*.

Mining Key Phrase Translations from Web Corpora

Fei Huang **Ying Zhang** **Stephan Vogel**

School of Computer Science
Carnegie Mellon University, Pittsburgh, PA 15213
{fhuang, joy, vogel}@cs.cmu.edu

Abstract

Key phrases are usually among the most information-bearing linguistic structures. Translating them correctly will improve many natural language processing applications. We propose a new framework to mine key phrase translations from web corpora. We submit a source phrase to a search engine as a query, then expand queries by adding the translations of topic-relevant hint words from the returned snippets. We retrieve mixed-language web pages based on the expanded queries. Finally, we extract the key phrase translation from the second-round returned web page snippets with phonetic, semantic and frequency-distance features. We achieve 46% phrase translation accuracy when using top 10 returned snippets, and 80% accuracy with 165 snippets. Both results are significantly better than several existing methods.

1 Introduction

Key phrases such as named entities (person, location and organization names), book and movie titles, science, medical or military terms and others [1], are usually among the most information-bearing linguistic structures. Translating them correctly will improve the performance of cross-lingual information retrieval, question answering and machine translation systems. However, these key phrases are often domain-specific, and people constantly create new key phrases which are not covered by existing bilingual dictionaries or parallel corpora, therefore standard data-driven or knowledge-based machine translation systems cannot translate them correctly.

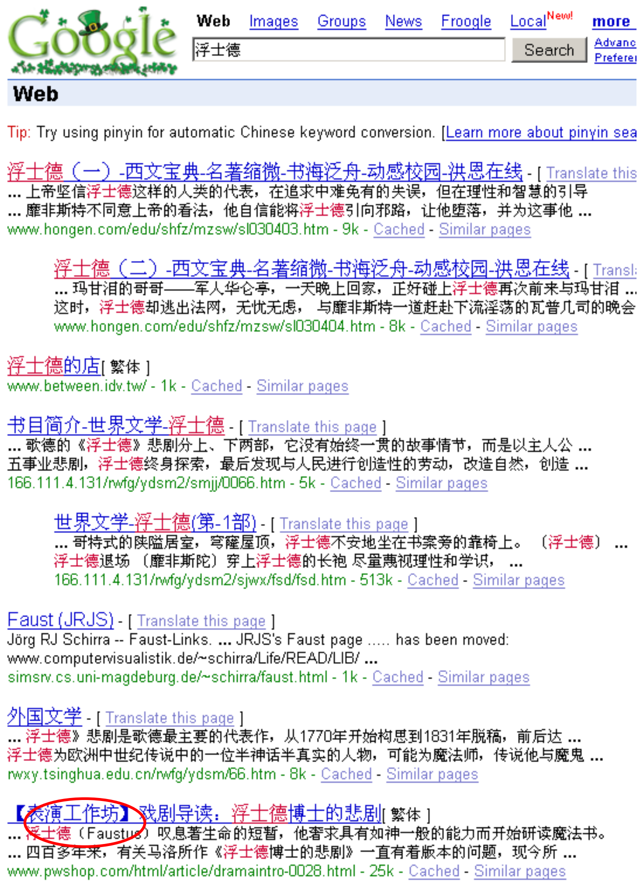
As an increasing amount of web information becomes available, exploiting such a huge information resource is becoming more attractive. (Resnik 1999) searched the web for parallel corpora while (Lu et al. 2002) extracted translation pairs from anchor texts pointing to the same webpage. However, parallel webpages or anchor texts are quite limited, and these approaches greatly suffer from the lack of data.

However, there are many web pages containing useful bilingual information where key phrases and their translations both occur. See the example in Figure 1. This example demonstrates web page snippets[2] containing both a Chinese key phrase "浮士德" and its translation**,** "Faust".

We thus can transform the translation problem into a data mining problem by retrieving these mixed-language web pages and extracting their translations. We propose a new framework to mine key phrase translations from web corpora. Given a source key phrase (here a Chinese phrase), we first retrieve web page snippets containing this phrase using the Google search engine. We then expand queries by adding the translations of topic-relevant hint words from the returned snippets. We submit the source key phrase and expanded queries again to Google to retrieve mixed-language web page snippets. Finally, we extract the key phrase translation from the second-round returned snippets with phonetic, semantic and frequency-distance features.

[1] Some name and terminology is a single word, which could be regarded as a one-word phrase.

[2] A snippet is a sentence or paragraph containing the key phrase, returned with the web page URLs.



浮士德

Tip: Try using pinyin for automatic Chinese keyword conversion. [Learn more about pinyin sea

浮士德（一）-西文宝典-名著缩微-书海泛舟-动感校园-洪恩在线 - [Translate this
... 上帝坚信浮士德这样的人类的代表，在追求中难免有的失误，但在理性和智慧的引导
... 靡非斯特不同意上帝的看法，他自信能将浮士德引向邪路，让他堕落，并为这事他 ...
www.hongen.com/edu/shfz/mzsw/sl030403.htm - 9k - Cached - Similar pages

浮士德（二）-西文宝典-名著缩微-书海泛舟-动感校园-洪恩在线 - [Transl
... 玛甘泪的哥哥——军人华仑亭，一天晚上回家，正好碰上浮士德再次前来与玛甘泪 ...
这时，浮士德却逃出法网，无忧无虑， 与靡非斯特一道赶赴下流淫荡的瓦普几司的晚会
www.hongen.com/edu/shfz/mzsw/sl030404.htm - 8k - Cached - Similar pages

浮士德的店[繁体]
www.between.idv.tw/ - 1k - Cached - Similar pages

书目简介-世界文学-浮士德 - [Translate this page]
... 歌德的《浮士德》悲剧分上、下两部，它没有始终一贯的故事情节，而是以主人公 ...
五事业悲剧，浮士德终身探索，最后发现与人民进行创造性的劳动，改造自然，创造 ...
166.111.4.131/rwfg/ydsm2/smjj/0066.htm - 5k - Cached - Similar pages

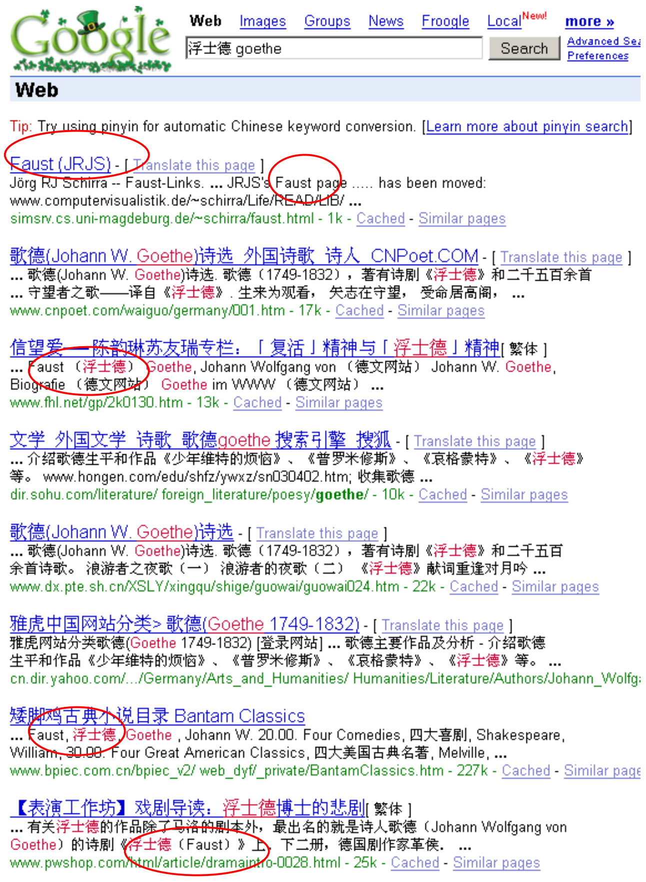
世界文学-浮士德(第-1部) - [Translate this page]
... 哥特式的陕隘居室，穹窿屋顶，浮士德不安地坐在书案旁的靠椅上。（浮士德） ...
浮士德退场 （靡非斯陀）穿上浮士德的长袍 尽量蔑视理性和学识， ...
166.111.4.131/rwfg/ydsm2/sjwx/fsd/fsd.htm - 513k - Cached - Similar pages

Faust (JRJS) - [Translate this page]
Jörg RJ Schirra -- Faust-Links. ... JRJS's Faust page ..... has been moved:
www.computervisualistik.de/~schirra/Life/READ/LIB/ ...
simsrv.cs.uni-magdeburg.de/~schirra/faust.html - 1k - Cached - Similar pages

外国文学 - [Translate this page]
... 浮士德》悲剧是歌德最主要的代表作，从1770年开始构思到1831年脱稿，前后达 ...
浮士德为欧洲中世纪传说中的一位半神话半真实的人物，可能为魔法师，传说他与魔鬼 ...
rwxy.tsinghua.edu.cn/rwfg/ydsm/66.htm - 8k - Cached - Similar pages

【表演工作坊】戏剧导读：浮士德博士的悲剧[繁体]
... 浮士德（Faustus）叹息著生命的短暂，他奢求具有如神一般的能力而开始研读魔法书。
... 四百多年来，有关马洛所作《浮士德博士的悲剧》一直有着版本的问题，现今所 ...
www.pwshop.com/html/article/dramaintro-0028.html - 25k - Cached - Similar pages

Figure 1. Returned mixed-language web page snippets using source query



浮士德 goethe

Tip: Try using pinyin for automatic Chinese keyword conversion. [Learn more about pinyin search]

Faust (JRJS) - [Translate this page]
Jörg RJ Schirra -- Faust-Links. ... JRJS's Faust page ..... has been moved:
www.computervisualistik.de/~schirra/Life/READ/LIB/ ...
simsrv.cs.uni-magdeburg.de/~schirra/faust.html - 1k - Cached - Similar pages

歌德(Johann W. Goethe)诗选 外国诗歌 诗人 CNPoet.COM - [Translate this page]
... 歌德(Johann W. Goethe)诗选. 歌德（1749-1832），著有诗剧《浮士德》和二千五百余首
... 守望者之歌——译自《浮士德》. 生来为观看， 矢志在守望， 受命居高阁， ...
www.cnpoet.com/waiguo/germany/001.htm - 17k - Cached - Similar pages

信望爱——陈韵琳苏友瑞专栏：「复活」精神与「浮士德」精神[繁体]
... Faust （浮士德） Goethe, Johann Wolfgang von （德文网站） Johann W. Goethe,
Biografie （德文网站） Goethe im WWW （德文网站） ...
www.fhl.net/gp/2k0130.htm - 13k - Cached - Similar pages

文学 外国文学 诗歌 歌德goethe 搜索引擎 搜狐 - [Translate this page]
... 介绍歌德生平和作品《少年维特的烦恼》、《普罗米修斯》、《哀格蒙特》、《浮士德》
等。 www.hongen.com/edu/shfz/ywxz/sn030402.htm; 收集歌德 ...
dir.sohu.com/literature/ foreign_literature/poesy/**goethe**/ - 10k - Cached - Similar pages

歌德(Johann W. Goethe)诗选 - [Translate this page]
... 歌德(Johann W. Goethe)诗选. 歌德（1749-1832），著有诗剧《浮士德》和二千五百
余首诗歌。 浪游者之夜歌（一） 浪游者的夜歌（二） 《浮士德》献词重逢对月吟 ...
www.dx.pte.sh.cn/XSLY/xingqu/shige/guowai/guowai024.htm - 22k - Cached - Similar pages

雅虎中国网站分类> 歌德(Goethe 1749-1832) - [Translate this page]
雅虎网站分类歌德(Goethe 1749-1832) [登录网站] ... 歌德主要作品及分析 - 介绍歌德
生平和作品《少年维特的烦恼》、《普罗米修斯》、《哀格蒙特》、《浮士德》等。 ...
cn.dir.yahoo.com/.../Germany/Arts_and_Humanities/ Humanities/Literature/Authors/Johann_Wolfg

矮脚鸡古典小说目录 Bantam Classics
... Faust, 浮士德, Goethe , Johann W. 20.00. Four Comedies, 四大喜剧, Shakespeare,
William, 30.00. Four Great American Classics, 四大美国古典名著, Melville, ...
www.bpiec.com.cn/bpiec_v2/ web_dyf/_private/BantamClassics.htm - 227k - Cached - Similar page

【表演工作坊】戏剧导读：浮士德博士的悲剧[繁体]
... 有关浮士德的作品除了马洛的剧本外，最出名的就是诗人歌德（Johann Wolfgang von
Goethe）的诗剧《浮士德（Faust）》上、下二册，德国剧作家革侯. ...
www.pwshop.com/html/article/dramaintro-0028.html - 25k - Cached - Similar pages

Figure 2. Returned mixed-language web page snippets using cross-lingual query expansion

We achieve 46% phrase translation accuracy when using 10 returned snippets, and 80% accuracy with 165 snippets. Both results are significantly better than several existing methods.

The reminder of this paper is organized as follows: cross-lingual query expansion is discussed in section 2; key phrase translation extraction is addressed in section 3. In section 4 we present experimental results, which is followed by relevant works and conclusions.

2 Retrieving Web Page Snippets through Cross-lingual Query Expansion

For a Chinese key phrase *f*, we want to find its translation *e* from the web, more specifically, from the mixed-language web pages or web page snippets containing both *f* and *e*. As we do not know *e*, we are unable to directly retrieve such mixed-language web page using (*f*,*e*) as the query.

However, we observed that when the author of a web page lists both *f* and *e* in a page, it is very likely that *f*′ and *e*′ are listed in the same page, where *f’* is a Chinese hint word topically relevant to *f*, and *e’* is *f*’s translation. Therefore if we know a Chinese hint word *f’*, and we know its reliable translation, *e’*, we can send (*f*, *e’*) as a query to retrieve mixed language web pages containing (*f*, *e*).

For example, to find web pages which contain translations of “浮士德”(Faust), we expand the query to “浮士德+goethe” since “歌德” (Goethe) is the author of “浮士德” (Faust). Figure 2 illustrates retrieved web page snippets with expanded queries. We find that newly returned snippets contain more correct translations with higher ranks.

To propose a “good” English hint *e*′ for *f*, first we need to find a Chinese hint word *f*′ that is relevant to *f*. Because *f* is often an OOV word, it is unlikely that such information can be obtained from existing Chinese monolingual corpora. Instead, we query Google for web pages containing *f*. From the returned snippets we select Chinese words *f'* based on the following criteria:

1. *f'* should be relevant to *f* based on the co-occurrence frequency. On average, 300 Chinese words are returned for each query *f*. We only consider those words that occur at least twice to be relevant.
2. *f'* can be reliably translated given the current bilingual resources (e.g. the LDC Chinese-English lexicon[3] with 81,945 translation entries).
3. The meaning of *f'* should not be too ambiguous. Words with many translations are not used.
4. *f'* should be translated into noun or noun phrases. Given the fact that most OOV words are noun or noun phrases, we ignore those source words which are translated into other part-of-speech words. The British National Corpus[4] is used to generate the English noun lists.

For each *f*, the top Chinese words *f'* with the highest frequency are selected. Their corresponding translations are then used as the cross-lingual hint words for *f*. For example, for OOV word *f* = 浮士德 (Faust), the top candidate *f'*s are "歌德(Goethe)", "简介(introduction)", "文学(literature)" and "悲剧(tragedy)". We expand the original query "浮士德" to "浮士德+ goethe", "浮士德 + introduction", "浮士德 + literature", "浮士德 + tragic", and then query Google again for web page snippets containing the correct translation "Faust".

3 Extracting Key Phrase Translation

When the Chinese key phrase and its English hint words are sent to Google as the query, returned web page snippets contain the source query and possibly its translation. We preprocess the snippets to remove irrelevant information. The preprocessing steps are:

1. Filter out HTML tags;
2. Convert HTML special characters (e.g., "&lt") to corresponding ASCII code (">");
3. Segment Chinese words based on a maximum string matching algorithm, which is used to calculate the translation probability between a Chinese key phrase and an English translation candidate.
4. Replace punctuation marks with phrase separator '|';
5. Replace non-query Chinese words with placeholder mark '+', as they indicate the distance between an English phrase and the Chinese key phrase.

For example, the snippet

> 《<b>廊桥遗梦</b>》(the bridges of madison county)[review]. 发布者：anjing | 发布时间：2004-01-25 星期日 02:13 | 最新更新时间

is converted into

> | <b> 廊 桥 遗 梦 </b> | the_bridges_of_Madison_county | review | ++ + | anjing | ++ ++ | 2004-01-25 +++ 02 13 | + + ++ ++,

where "<b>" and "</b>" mark the start and end positions of the Chinese key phrase. The candidate English phrases, "the bridges of madison county", "review" and "anjing", will be aligned to the source key phrase according to a combined feature set using a transliteration model which captures the pronunciation similarity, a translation model which captures the semantic similarity and a frequency-distance model reflecting their relevancy. These models are described below.

3.1 Transliteration Model

The transliteration model captures the phonetic similarity between a Chinese phrase and an English translation candidate via string alignment. Many key phrases are person and location names, which are phonetically translated and whose written forms resemble their pronunciations. Therefore it is possible to discover these translation pairs through their surface strings. Surface string transliteration does not need a pronunciation lexicon to map words into phoneme sequences; thus it is especially appealing for OOV word translation. For non-Latin languages like Chinese, a romanization

[3]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L27
[4] http://www.natcorp.ox.ac.uk/

script called "pinyin" maps each Chinese character into Latin letter strings. This normalization makes the string alignment possible.

We adopt the transliteration model proposed in (Huang, et al. 2003). This model calculates the probabilistic Levinstein distance between a romanized source string and a target string. Unlike the traditional Levinstein distance calculation, the character alignment cost is not binary (0/1); rather it is the logarithm of character alignment probability, which ensures that characters with similar pronunciations (e.g. `p` and `b`) have higher alignment probabilities and lower cost. These probabilities are automatically learned from bilingual name lists using EM.

Assume the Chinese phrase f has J Chinese characters, $f_1, f_2, ... f_J$, and the English candidate phrase e has L English words, $e_1, e_2, ..., e_L$. The transliteration cost between a Chinese query f and an English translation candidate e is calculated as:

$$C_{trl}(e, f) \approx \sum_j \log p(e_{a_j} \mid y_j) = \sum_j \sum_i \log p(e_{a_{(i,j)}} \mid y_{i,j}).$$

where y_j is the pinyin of Chinese character f_j, $y_{j,i}$ is the i th letter in y_j, and e_{a_j} and $e_{a_{(j,i)}}$ are their aligned English letters, respectively. $p(e_{(i,j)} \mid y_{i,j})$ is the letter transliteration probability. The transliteration costs between a Chinese phrase and an English phrase is approximated by the sum of their letter transliteration cost along the optimal alignment path, which is identified based on dynamic programming.

3.2 Translation Model

The translation model measures the semantic equivalence between a Chinese phrase and an English candidate. One widely used model is the IBM model (Brown et al. 1993). The phrase translation probability is computed using the IBM model-1 as:

$$P_{trans}(f \mid e) = \frac{1}{L^J} \prod_{j=1}^{J} \sum_{l=1}^{L} p(f_j \mid e_l)$$

where $p(f_j \mid e_l)$ is the lexical translation probabilities, which can be calculated according to the IBM models. This alignment model is asymmetric, as one source word can only be aligned to one target word, while one target word can be aligned to multiple source words. We estimate both $P_{trans}(f \mid e)$ and $P_{trans}(e \mid f)$, and define the NE translation cost as:

$$C_{trans}(e, f) = \log P_{trans}(e \mid f) + \log P_{trans}(f \mid e).$$

3.3 Frequency-Distance Model

The more often a bilingual phrase pair co-occurs, or the closer a bilingual phrase pair is within a snippet, the more likely they are translations of each other. The frequency-distance model measures this correlation.

Suppose S is the set of returned snippets for query f, and a single returned snippet is $s_i \in S$. The source phrase occurs in s_i as $f_{i,j}$ ($j \geq 1$ since f may occur several times in a snippet). The frequency-distance weight of an English candidate e is

$$w(e) = \sum_{s_i} \sum_{f_{i,j}} \frac{1}{d(f_{i,j}, e)}$$

where $d(f, e)$ is the distance between phrase $f_{i,j}$ and e, i.e., how many words are there between the two phrases (the separator `|` is not counted).

3.4 Feature Combination

Define the confidence measure for the transliteration model as:

$$\phi_{trl}(e \mid f) = \frac{\exp[C_{trl}(e, f)] w(e)^m}{\sum_{e'} \exp[C_{trl}(e', f)] w(e')^m},$$

where *e* and *e'* are English candidate phrases, and *m* is the weight of the distance model. We empirically choose m=2 in our experiments. This measure indicates how good the English phrase *e* is compared with other candidates based on transliteration model. Similarly the translation model confidence measure is defined as:

$$\phi_{trans}(e \mid f) = \frac{\exp[C_{trans}(e, f)] w(e)^m}{\sum_{e'} \exp[C_{trans}(e', f)] w(e')^m}.$$

The overall feature cost is the linear combination of transliteration cost and translation cost, which are weighted by their confidence scores respectively:

$$C(e,f)=\lambda\phi_{trl}(e\mid f)\exp[C_{trl}(e,f)]+ (1-\lambda)\phi_{trans}(e\mid f)\exp[C_{trans}(e,f)]$$

where the linear combination weight λ is chosen empirically. While ϕ_{trl} and ϕ_{trans} represent the relative rank of the current candidate among all compared candidates, C_{trl} and C_{trans} indicate its absolute likelihood, which is useful to reject the top 1 incorrect candidate if the true translation does not occur in any returned snippets.

4 Experiments

We evaluated our approach by translating a set of key phrases from different domains. We selected 310 Chinese key phrases from 12 domains as the test set, which were almost equally distributed within these domains. We also manually translated them as the reference translations. Table 1 shows some typical phrases and their translations, where one may find that correct key phrase translations need both phonetic transliterations and semantic translations. We evaluated *inclusion rate*, defined as the percentage of correct key phrase translations which can be retrieved in the returned snippets; *alignment accuracy*, defined as the percentage of key phrase translations which can be correctly aligned given that these translations are included in the snippets; and *overall translation accuracy*, defined as the percentage of key phrases which can be translated correctly. We compared our approach with the LiveTrans[5] (Cheng et.al. 2004) system, an unknown word translator using web corpora, and we observed better translation performance using our approach.

4.1 Query Translation Inclusion Rate

In the first round query search, for each Chinese key phrase *f*, on average 13 unique snippets were returned to identify relevant Chinese hint words *f'*, and the top 5 *f*'s were selected to generate hint words *e*'s. In the second round *f* and *e*'s were sent to Google again to retrieve mixed language snippets, which were used to extract *e*, the correct translation of *f*.

Figure 3 shows the inclusion rate vs. the number of snippets used for three mixed-language web page searching strategies:

[5] http://livetrans.iis.sinica.edu.tw/lt.html

Movie Title	廊桥遗梦 the Bridges of Madison-County 阿甘正传 Forrest Gump
Book Title	红楼梦 Dream of the Red Mansion 茶花女 La Dame aux camellias
Organization Name	圣母大学 University of Notre Dame 大卫与露西派克德基金会 David and Lucile Packard Foundation
Person Name	贝多芬 Ludwig Van Beethoven 奥黛丽赫本 Audrey Hepburn
Location Name	勘察加半岛 Kamchatka 塔克拉玛干沙漠 Taklamakan desert
Company / Brand	汉莎航空 Lufthansa German Airlines 雅诗兰黛 Estee Lauder
Sci&Tech Terms	遗传算法 genetic algorithm 语音识别 speech recognition
Specie Term	秃鹫 Aegypius monachus 穿山甲 Manispentadactyla
Military Term	宙斯盾 Aegis 费尔康 Phalcon
Medical Term	非典型性肺炎 SARS 动脉硬化 Arteriosclerosis
Music Term	空山鸟语 Bird-call in the Mountain 巴松管 Bassoon
Sports Term	休斯敦火箭队 Houston Rockets 环法自行车赛 Tour de France

Table 1. Test set key phrases

- Search any web pages containing *f* (Zhang and Vines 2004);
- Only search *English* web pages[6] containing *f* (Cheng et al. 2004);
- Search any web pages containing *f* and hint words *e'*, as proposed in this paper.

The first search strategy resulted in a relatively low inclusion rate; the second achieved a much higher inclusion rate. However, because such English pages were limited, and on average only 45 unique snippets could be found for each *f*, which resulted in a maximum inclusion rate of 85.8%. In the case of the cross-lingual query expansion, the search space was much larger but more focused and we achieved a high inclusion rate of 89.7% using 32 mixed-language snippets and 95.2% using 165 snippets, both from the second round retrieval.

[6] These web pages are labeled by Google as "English" web pages, though they may contain non-English characters.

Features	No Hints (Inc = 44.19%) (avg. snippets = 10)	With Hints (Inc = 95.16%) (avg. snippets=130)
Trl	51.45	17.97
Trans	51.45	40.68
Fq-dis	53.62	73.22
Trl+Trans	63.04	51.36
Trl+Trans+ Fq-dis	65.94	86.73

Table 2. Alignment accuracies using different features

These search strategies are further discussed in the section 5.

4.2 Translation Alignment Accuracy

We evaluated our key phrase extraction model by testing queries whose correct translations were included in the returned snippets. We used different feature combinations on differently sized snippets to compare their alignment accuracies. Table 2 shows the result. Here "Trl" means using the transliteration model, "Trans" means using the translation model, and "Fq-dis" means using Frequency-Distance model. The frequency-distance model seemed to be the strongest single model in both cases (with and without hint words), while incorporating phonetic and semantic features provided additional strength to the overall performance. Combining all three features together yielded the best accuracy. Note that when more candidate translations were available through query expansion, the alignment accuracy improved by 30% relative due to the frequency-distance model. However, using transliteration and/or translation models alone decreased performance because of more incorrect translation candidates from returned snippets. After incorporating the frequency-distance model, correct translations have the maximum frequency-distance weights and are more likely to be selected as the top hypothesis. Therefore the combined model obtained the highest translation accuracy.

4.3 Overall Translation Quality

The overall translation qualities are listed in Table 3, where we showed the translation accuracies of the top 5 hypotheses using different number of snippets. A hypothesized translation was considered to be correct when it matched one of the reference translations. Using more snippets always increased the overall translation accuracy, and with all the 165 snippets (on average per query), our approach achieved 80% top-1 translation accuracy, and 90% top-5 accuracy.

Snippets Used	Accuracy of the Top-N Hyp. (%)				
	Top1	**Top2**	**Top3**	**Top4**	**Top5**
10	46.1	55.2	59.0	61.3	62.3
20	57.4	64.2	69.7	72.6	72.9
50	63.2	74.5	77.7	79.7	80.6
100	75.2	84.5	85.8	87.4	87.4
165	**80.0**	**86.5**	**89.0**	**90.0**	**90.0**
BabelFish[7] MT	31.3	N/A	N/A	N/A	N/A
CMU-SMT	21.9	N/A	N/A	N/A	N/A
LiveTrans (Fast)	20.6	30.0	36.8	41.9	45.2
LiveTrans (Smart)	30.0	41.9	48.7	51.0	52.9

Table 3. Overall translation accuracy

We compared the translations from a research statistical machine translation system (CMU-SMT, Vogel et al. 2003) and a web-based MT engine (BabelFish). Due to the lack of topic-relevant contexts and many OOV words occurring in the source key phrases, their results were not satisfactory. We also compare our system with LiveTrans, which only searched within English web pages, thus with limited search space and more noises (incorrect English candidates). Therefore it was more difficult to select the correct translation. Table 4 lists some example key phrase translations mined from web corpora, as well as translations from the BabelFish.

5 Relevant Work

Both (Cheng et al. 2004) and (Zhang and Vines 2004) exploited web corpora for translating OOV terms and queries. Compared with their work, our proposed method differs in both webpage search

[7] http://babelfish.altavista.com/

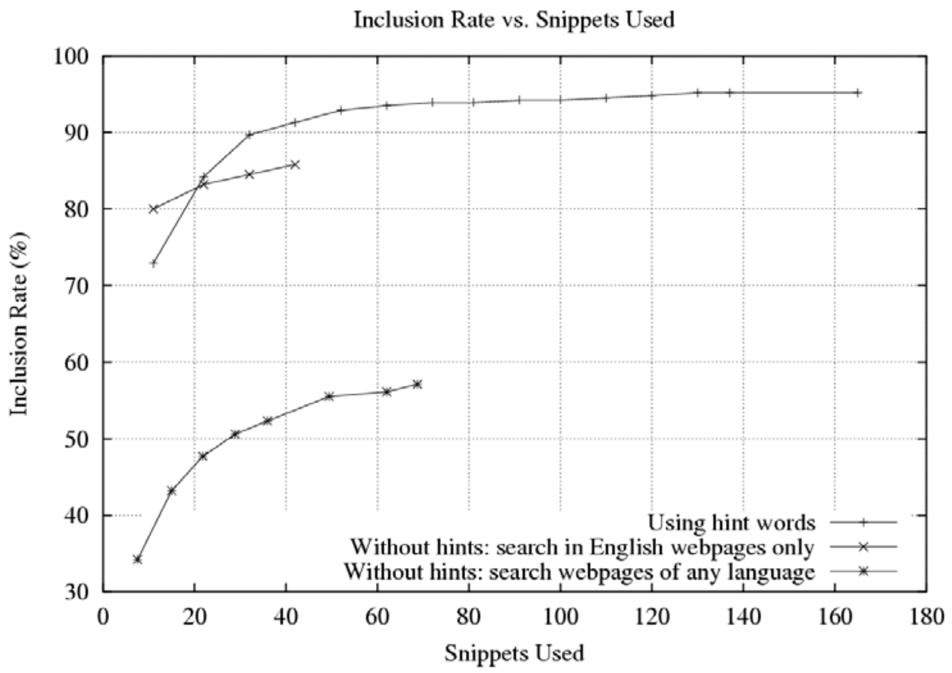


Figure 3. Inclusion rate vs. the number of snippets used

Category	Examples		
	Chinese Key Phrase	**Web-Mining Translation**	**BabelFish™ Result**
Movie Title	廊桥遗梦	the Bridges of Madison County	*Love has gone and only good memory has left in the dream
Book Title	理智与情感	Sense and Sensibility	*Reason and emotion
Organization Name	伍德威尔逊全国联谊基金会	Woodrow Wilson National Fellowship Foundation	*Wood the Wilson nation gets together the foundation
Person Name	小泽征尔	Seiji Ozawa	*Young Ze drafts you
Location Name	柴达木盆地	Tsaidam Basin	Qaidam Basin
Company / Brand	倩碧	Clinique	*Attractive blue
Sci&Tech Terms	贝叶斯网络	Bayesian network	*Shell Ye Si network
Specie Term	海象	walrus	walrus
Military Term	同温层堡垒	stratofortress	stratofortress
Medical Term	青光眼	glaucoma	glaucoma
Music Term	巴松管	bassoon	bassoon
Sports Term	环法自行车赛	Km Tour de France	*Link law bicycle match

*: Incorrect translations

Table 4. Key phrase translation from web mining and a MT engine

space and translation extraction features. Figure 4 illustrates three different search strategies. Suppose we want to translate the Chinese query “浮士德”. (Cheng et al. 2004) only searched 188 English web pages which contained the source query, and 53% of them (100 pages) had the correct translations. (Zhang and Vines 2004) searched the whole 55,100 web pages, 10% of them (5490 pages) had the correct translation. Our approach used query expansion to search any web pages containing “浮士德” and English hint words, which was a larger search space than (Cheng et al. 2004) and more focused compared with (Zhang and Vines 2004), as illustrated with the shaded region in Figure 4. For translation extraction features, we took advantage of machine transliteration and machine translation models, and combined them with frequency and distance information.

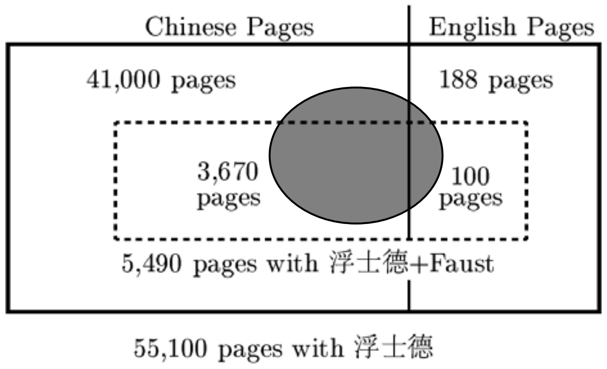


Figure 4. Web search space strategy comparison

6 Discussion and Future Work

In this paper we demonstrated the feasibility of the proposed approach by searching for the English translation for a given Chinese key phrase, where we use punctuations and Chinese words as the boundary of candidate English translations. In the future we plan to try more flexible translation candidate selection methods, and apply them to other language pairs. We also would like to test our approach on more standard test sets, and compare the performance with other systems.

Our approach works on short snippets for query expansion and translation extraction, and the computation time is short. Therefore the search engine’s response time is the major factor of computational efficiency.

7 Conclusion

We proposed a novel approach to mine key phrase translations from web corpora. We used cross-lingual query expansion to retrieve more relevant web pages snippets, and extracted target translations combining transliteration, translation and frequency-distance models. We achieved significantly better results compared to the existing methods.

8 References

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra and R.L. Mercer. The Mathematics of Machine Translation: Parameter Estimation. *In Computational Linguistics*, vol 19, number 2. pp.263-311, June, 1993.

P.–J. Cheng, J.-W. Teng, R.-C. Chen, J.-H. Wang, W.-H. Lu, and L.-F. Chien. Translating unknown queries with web corpora for cross-language information retrieval. In the Proceedings of 27th ACM SIGIR, pp146-153. ACM Press, 2004.

F. Huang, S.Vogel and A. Waibel. Automatic extraction of named entity translingual equivalence based on multi-feature cost minimization. In the Proceedings of the 41st ACL. Workshop on Multilingual and Mixed-language Named Entity Recognition, pp124-129, Sapporo, Japan, July 2003.

W.-H. Lu, L.-F. Chien, H.-J. Lee. Translation of web queries using anchor text mining. ACM Trans. Asian Language Information Processing (TALIP) 1(2): 159-172 (2002)

P. Resnik and N. A. Smith, The Web as a Parallel Corpus, Computational Linguistics 29(3), pp. 349-380, September 2003

S. Vogel, Y. Zhang, F. Huang, A. Tribble, A. Venogupal, B. Zhao and A. Waibel. The CMU statistical machine translation system. *In Proceedings of the MT Summit IX Conference* New Orlean, LA, September, 2003.

Y. Zhang and P. Vines. Detection and Translation of OOV Terms Prior to Query Time, In the Proceedings of 27th ACM SIGIR. pp524-525, Sheffield, England, 2004.

Y. Zhang and P. Vines 2004, Using the Web for Automated Translation Extraction in Cross-Language Information Retrieval, In Proceedings of 27th ACM SIGIR, pp.162-169, Sheffield, United Kingdom, 2004.

Y. Zhang, F. Huang and S. Vogel, Mining Translations of OOV Terms from the Web through Cross-lingual Query Expansion, in the Proceedings of the 28th ACM SIGIR, Salvador, Brazil, August 2005.

Robust Named Entity extraction from large spoken archives

Benoît Favre
Thales, MMP Laboratory
Colombes, France
benoit.favre@
fr.thalesgroup.com

Frédéric Béchet
LIA, University of Avignon
Avignon, France
frederic.bechet@
univ-avignon.fr

Pascal Nocéra
LIA, University of Avignon
Avignon, France
pascal.nocera@
univ-avignon.fr

Abstract

Traditional approaches to Information Extraction (IE) from speech input simply consist in applying text based methods to the output of an Automatic Speech Recognition (ASR) system. If it gives satisfaction with low Word Error Rate (WER) transcripts, we believe that a tighter integration of the IE and ASR modules can increase the IE performance in more difficult conditions. More specifically this paper focuses on the robust extraction of Named Entities from speech input where a temporal mismatch between training and test corpora occurs. We describe a Named Entity Recognition (NER) system, developed within the French Rich Broadcast News Transcription program ESTER, which is specifically optimized to process ASR transcripts and can be integrated into the search process of the ASR modules. Finally we show how some *metadata* information can be collected in order to adapt NER and ASR models to new conditions and how they can be used in a task of Named Entity indexation of spoken archives.

1 Introduction

Named Entity Recognition (NER) is a crucial step in many Information Extraction (IE) tasks. It has been a specific task in several evaluation programs such as the Message Understanding Conferences (MUC), the Conferences on Natural Language Learning (CoNLL), the DARPA HUB-5 program or more recently the French ESTER Rich Transcription program on Broadcast News data. Most of these conferences have studied the impact of using transcripts generated by an Automatic Speech Recognition (ASR) system rather than written texts. It appears from these studies that unlike other IE tasks, NER performance is greatly affected by the Word Error Rate (WER) of the transcripts processed. To tackle this problem, different ideas have been proposed: modeling explicitly the ASR errors (Palmer and Ostendorf, 2001) or using the ASR system alternate hypotheses found in word lattices (Saraclar and Sproat, 2004). However performance in NER decreases dramatically when processing high WER transcripts like the ones that are obtained with unmatched conditions between the ASR training model and the data to process. This paper investigates this phenomenon in the framework of the NER task of the French Rich Transcription program of Broadcast News ESTER (Gravier et al., 2004). Several issues are addressed:

- how to jointly optimize the ASR and the NER models ?
- what is the impact in term of ASR and NER performance of a temporal mismatch between the corpora used to train and test the models and how can it be recovered by means of metadata information ?
- Can metadata information be used for indexing large spoken archives ?

After a quick overview of related works in IE from speech input, we present the ESTER evaluation program ; then we introduce a NER system tightly integrated to the ASR process and show how it can successfully index high WER spoken databases thanks to metadata.

2 Information extraction from large spoken archives

The NIST Topic Detection and Tracking (Fiscus and Doddington, 2002) and TREC document retrieval evaluation programs has studied the impact of recognition errors in the overall performance of Information Extraction systems for tasks like story segmentation or topic detection and retrieval. This impact has been shown to be very limited compared to clean text corpora. The main explanation for this phenomenon is the *redundancy effect*: themes, topics are very likely to be represented in texts by many occurrences of salient words characterizing them. Therefore, even if some of these words are missing, numerical Information Extraction methods can use the remaining salient words and discard the noise generated by ASR errors.

However, this phenomenon is not true for tasks related to the extraction of fine grained entities, like Named Entities. Indeed, several studies have shown that F-measure and WER are strongly correlated : 0.7 points of F-measure lost for each additional 1% of WER according to (Miller et al., 2000) on the experiments of 1998 NIST Hub-4 evaluations (Przybocki et al., 1999).

Despite the continuous improvement of ASR techniques, high WER transcriptions are inevitable in difficult conditions like those found in large spoken archives like in the MALACH project (Ramabhadran et al., 2003). Moreover, Named Entities extraction performance is greatly affected by a mismatch between training and testing data. This is due mainly because proper names, which represent most of the Named Entity items, are a very dynamic category of words, strongly related to the period of time representing the documents to process. Therefore this mismatch is inevitable when dealing with archives spreading over a long period of time and containing multiple domain information.

One way of tackling this problem is to gather *metadata* information related to the documents to process. This information can be newspaper corpora related to the same period of time, abstract describing the document content, or simply lists of terms or entities likely to occur. Although such collected data can be used to update the ASR and NER models, the potential gain is rather small unless the metadata corpus gathered fits perfectly the document to process and is of a reasonable size. But another way of exploiting this metadata information is to use it as set of index terms that are going to be explicitly looked for in the processed documents. We present in section 7 an implementation of this idea that uses word lattices as input.

3 The ESTER Named Entity evaluation program

This work has been done within the framework of the French Rich Transcription program of Broadcast News ESTER. ESTER is organized by *l'Association Francophone de la Communication Parlée* (AFCP), *la Délégation Générale pour l'Armement* (DGA) and the *Evaluation language Resources Distribution Agency* (ELDA). The ESTER corpus is made of 100 hours of Broadcast News data (from 6 French speaking radio channels), manually transcribed, and labeled with a tagset of about 30 Named Entity categories folded in 8 main types:

- persons (**pers**): human beings, fiction characters, animals;
- locations (**loc**): geographical, traffic lines, electronic and real addresses, dial numbers;
- organizations (**org**): political, business, non profit;
- geo-socio-political groups (**gsp**): clans, families, nations, administrative regions;
- amounts (**amount**): durations, money, lengths, temperature, age, weight and speed;
- time (**time**): relative and absolute time expressions, hours;
- products (**prod**): art, printings, awards and vehicles;

- facilities (**fac**): buildings, monuments.

This data is divided in 3 sets: a training set (84%), a development set(8%) and a test set (8%). There is a 6 month gap difference between the training corpus and the test corpus while the development corpus matches the training data from a temporal point of view: the training corpus contains Broadcast News spreading from 2002 to December 2003; the development corpus contains news from 2003; the test corpus has been recorded in October 2004. There are also 2 new radio channels in the test corpus which were not in the training data.

For these reasons the development data is called the *matched corpus* as the recording conditions match those of the training corpus and the test data is called the *unmatched corpus*. As a consequence, we can study the effect of unmatched conditions on ASR as well as IE performance and propose solutions for dealing with this problem.

One of the main characteristics of the ESTER corpus is the size of the NE tagset and the high ambiguity rate among the NE categories (eg. administrative regions and geographical locations): 83% of the *matched corpus* entities occur in the training corpus and 40% of them are ambiguous whereas only 61% of the *unmatched corpus* entities occur in the training corpus and 32% of them are ambiguous.

The most commonly used measures for evaluating NE extraction performance are Slot Error Rate (SER) and F-measure. SER is very similar to WER because it takes into account fine grained errors like insertions, deletions and substitutions (entity type and extent). The scoring process is based on the same alignment between reference and hypothesis data than the one obtained for measuring WER and SER is known for being more accurate and penalizing than F-measure. Both measures weights can be adjusted to favor recall or precision and therefore adapted to a specific task.

$$SER = 100 * \frac{\sum_{e \in \mathcal{E}} \alpha_e |e|}{|\textit{Ref slots}|} \qquad F_\beta = \frac{(1+\beta^2)RP}{R+\beta^2 P}$$

$$R = \frac{|\textit{Correct slots}|}{|\textit{Ref slots}|} \qquad P = \frac{|\textit{Correct slots}|}{|\textit{Hyp slots}|}$$

with $e \in \mathcal{E}$ being an error type (insertion, deletion, type, extent, type+extent, multiple) and α_e its weight (resp. $1, 1, .5, .5, .8, 1.5$) ; P is the precision and R the recall; F_1 is used in this paper.

4 Extracting NE from written text vs. ASR output

As previously mentioned in section 2, WER and SER performance are strongly correlated. Besides the intrinsic difficulties of ASR (robustness to noise, speaker variation, lack of coverage of the Language Models used, …), there is a source of errors which is particularly important in IE from speech input: the Out-Of-Vocabulary (OOV) word phenomenon. Indeed, ASR models are built on huge textual corpus and only contain the most frequent words to limit computation and memory usage. If this is the right approach to WER reduction, it is certainly not valuable to information extraction where unlikely events are considered as important. For instance, many document retrieval models use inverse document frequency (rareness) as a word weighting parameter. So, unlikely proper names are not in reach of the ASR transcription system and hence cannot be spotted by a Named Entity extraction module.

In addition to Out-of-Vocabulary words, two other phenomenons have also a strong impact on NER performance: the insertion of erroneous proper names that automatically trigger the insertion of an entity and spontaneous speech phenomenons. These speech dysfluencies (hesitations, filled pauses, false starts...) reduce the quality of the transcript because they are usually not covered by language models (built from textual data) or artificially introduced. One should remove these from the transcript to improve the quality of the labeling.

In order to deal with ASR errors two approaches have been proposed:

- modeling explicitly the ASR errors, thanks to a development corpus and a set of confidence measures, in order to detect the possible errors of the 1-best word string hypothesis (with the type of errors) before extracting the NEs (Palmer and Ostendorf, 2001);

- exploiting a search space bigger than the 1-best hypothesis alone, either by taking into account an n-best list (Zhai et al., 2004) or the whole word lattice (Saraclar and Sproat, 2004).

The method proposed in this paper is close to this second approach where the whole word lattice output by the ASR system is used in order to increase NER performance from noisy input.

We will present also in the next section a new strategy for adapting NER models to ASR transcripts, based on one of the main characteristics of such transcripts: a closed vocabulary is used by the ASR system. To our knowledge this has never been fully exploited by NER systems. Indeed while the key point of NER systems on written text is their generalization capabilities when processing unknown words, this feature is not relevant for ASR transcripts as the system cannot generate words out of the lexicon (there are no unknown words). Therefore we propose here to fully exploit this constraint (close vocabulary): since the OOV words cannot appear in the ASR transcripts, the NER models can by over-trained on the words belonging to the ASR lexicon. This is going to be developed in the next section.

5 Robust Named Entity extraction

We have developed in this study two NER systems: one is based on the freely available NLP tool *Lingpipe*[1], adapted and trained on the French ESTER corpus and dedicated to process text input. This system is going to be called NER_{text} in the experiment section. The second NER system has been developed for this study and is specifically built for being tightly integrated with the ASR processes. The two main features of this system, called NER_{asr} in the following, are its ability to process word lattices and the fact that the NER models are trained for a specific ASR lexicon. These two systems are going to be presented in the next sections.

5.1 Text-based NER system: NER_{text}

Among all the different methods that have been proposed for NER, one can find rule based models (Cunningham et al., 2002), Maximum Entropy models (Brothwick et al., 1998), Conditionnal Random Fields or probabilistic HMM-based models (Bikel et al., 1999).

Lingpipe implements an HMM-based model. It maximizes the probability of a tag sequence T_i over a word sequence W_i. A context of two preceding words and one preceding tag is used to approximate this probability. Generalization is done through a simple process: words occurring with low frequency are replaced by feature based categories (capitalized, contains digits, ...). In this approach, there must be one tag per word. Words starting and ending entities are labeled with special tags. Because some features are lacking in ASR transcripts (e.g. capitalization, digits, sentence boundaries, ...) some word lists for each kind of features are added as presented in (Appelt and Martin, 1999).

[1]Lingpipe: `http://alias-i.com/lingpipe/`

5.2 ASR-based NER system: NER_{asr}

Errors occurring in ASR output lead NER systems to overgenerate NE detections. This is due to both erroneous words insertions in the ASR transcripts as well as some abusive generalization made by the NER systems. If these generalization capabilities are very important for processing unknown words in written texts, they can be an handicap in a closed-vocabulary situation like the one observed when processing ASR output. In order to reduce and control the insertion rate of our NER system, we implemented a two level approach: the first level is made of NE grammars coded as Finite State Machine (FSM) transducers and the second level is a statistical HMM-based tagger.

5.2.1 NE transducers

To each NE category is attached a set of regular grammars, extracted from the ESTER training corpus and generalized thanks to the annotation guidelines and web-gathered word lists. Theses grammars are represented by Finite State Machines (FSMs) (thanks to the AT&T GRM/FSM toolkit (Allauzen et al., 2003)). These FSMs are transducers that accept word sequences on the input symbols and output NE labels on the output symbols. They are all grouped together in a single transducer, called T_{gram}, with a filler model that accepts any string of words. Because these FSMs are lexicalized with the words of the ASR lexicon, one can control the generalization capabilities of the grammars thanks to the occurrence contexts of these words in the training corpus. During the NER process, the first step is to compose the FSM representing the NE transducer and the output of the ASR module (either a 1-best word string or a word lattice, both encoded as an FSM called G).

5.2.2 NE tagger

The result of the composition of the NE transducer with the ASR output is an FSM ($G \circ T_{gram}$) containing all the possible parsing made by the NE grammars. In order to find the best analysis a statistical model is used to decide between entity types and entities boundaries. This model is a 2nd order n-gram model (trigram) represented by a weighted FSM (called T_{tagger}) with the same framework as the grammars. The most likely NE label sequence is obtained by finding the best path in the FSM: $G \circ T_{gram} \circ T_{tagger}$. This corresponds to maximize the following probability:

$$P_W = \prod_{i=1}^{n} P(W_i, T_i | W_{i-1}, T_{i-1}, W_{i-2}, T_{i-2})$$

This model is similar to the one implemented in *Lingpipe* but it uses different smoothing methods. Similarly, first and last words of entities are represented by special tags (this helps getting more accurate boundaries) and low frequency words (appearing less than a fixed number of times in the training corpus) are generalized using their Part-Of-Speech tags. The key points of this approach are that it has a better control of the generalization capabilities than a feature based NER system, thanks to the NE grammars; it integrates the closed vocabulary constraint of the ASR systems; and it is not limited to the 1-best word hypothesis but can use the full ASR search space (through word lattices) in order to detect entities. Processing word lattices allows us to output, at the end of the extraction process, an n-best list of NE hypotheses. To each hypothesis are attached two scores:

- the likelihood score given by the ASR model to the best word string supporting this NE hypothesis in the word lattice;
- the probability $P(W_n, T_n, \ldots, W_0, T_0)$ given by the NE tagger to the sequence of NE labels $T_0, \ldots, T_n$ and the sequence of words $W_0, \ldots, W_n$.

From this n-best list we can estimate the *Oracle* performance of the NER system. This measure is the recall measure upper bound than can be obtained by extracting all the possible entities from a word lattice, thanks to the NE transducers, and simulating a perfect strategy that always take the right decision in choosing among all the possible entities.

Decision strategies on such an n-best of NE hypothesis can also involve other levels of information on the document to process like the date or the theme, for example. In the evaluation presented in the next section we compare this Oracle performance measure to the results of the simplest decision strategy which consists in choosing the NE hypothesis with the highest likelihood.

5.3 Evaluation

The evaluation presented in Tables 1 and 2 is performed using the Slot Error Rate and the F-measure on the *matched* and *unmatched* corpora presented in section 3.

corpus	*matched*		*unmatched*	
tagger	SER	F-m	SER	F-m
NER$_{text}$	21	84	27	79
NER$_{asr}$	23	84	37	74
WER	0		0	

Figure 1: F-measure and Slot Error Rate measures on the ESTER reference corpora (*matched* and *unmatched*) for both NER systems

corpus	*matched*		*unmatched*		
tagger	SER	F-m	SER	F-m	Oracle
NER$_{text}$	42	72	55	63	61.9
NER$_{asr}$	41	73	54	63	76.9
WER	21.2		26.4		

Figure 2: F-measure, Slot Error Rate and Oracle recall measures on the ASR output of the *matched* and *unmatched* corpora for both NER systems

Figure 1 presents SER and F-measure on the two test sets (*matched* and *unmatched*) for the text oriented (*NER$_{text}$*) and the speech oriented (*NER$_{asr}$*) NER systems, on clean text (manual transcripts). Figure 2 shows the results obtained on the ASR transcripts.

As expected on manually transcribed data, *NER$_{text}$* obtains better results than *NER$_{asr}$* (which has poorer generalization capabilities). On the ASR outputs the results obtained by both systems are comparable however *NER$_{asr}$* has the advantage of processing word lattices, leading to an interesting Oracle performance. We are studying now more elaborate decision strategies in order to take fully advantage of this feature.

The decrease in F-measure observed between the reference and the ASR transcripts is similar to the one obtained in other studies (Miller et al., 2000). One observation that can be made on these results is the impact of the time mismatch between the training and the test corpora. A 6 month difference in the *unmatched* corpus leads to a very big drop in both SER and F-measure. This can be explained by the fact that NEs are very time-dependent. We are going now to present some methods developed to tackle this problem.

6 Updating Language and NE models with metadata information

The only mismatch between the training and the *unmatched* corpus of our experiments is a 6 months temporal mismatch, therefore we collected a corpus of newsletters made on a daily basis by the French newspaper *Le Monde* corresponding to these 6 months. These newsletters contain an abstract of the news of each day. We make the following two hypotheses:

- firstly these newsletters are related to the same time period as the *unmatched* corpus, therefore integrating them into the ASR models (lexicon+Language Model) should help reducing the OOV word effect;
- secondly because they represent an abstract of the news of each day, the Named Entities occurring in a particular newsletter should contain all the major events of the corresponding day and therefore constitutes a useful list of terms that can be used for indexing a Broadcast News document related to the same period of time.

6.1 NE distribution analysis

This newsletter corpus contains 1M words and after being tagged by the *NER$_{text}$* system, 140k entities were extracted. To check the relevance of this corpus for adapting the models to the *unmatched* test corpus, we studied the distribution of the words and the entities for each day, from January to December 2004. The *unmatched* test corpus is made of Broadcast News ranging from October 10th to October 16th 2004. The following observations were made: 72% of the NEs and 60% of the words contained in them occur only one day in this corpus; the intersection of the NEs occurring in both the newsletter of a particular day and the entities belonging to the *unmatched* test corpus shows a peak, illustrated by figure 3, for the days of the test corpus; at this peak, 25% of the NEs are used the same day in the two corpora.

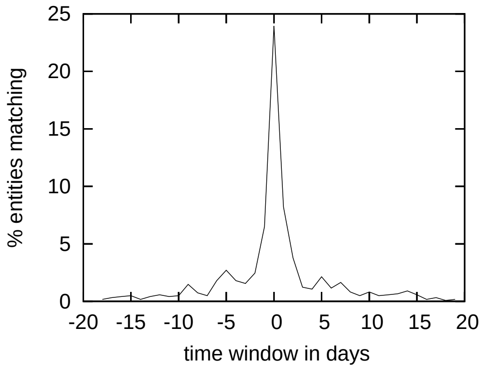


Figure 3: Percentage of entities of the unmatched corpus occuring at least n days earlier or later in the newsletter corpus (at a window of 0 days, entities appear on the same day in both corpus).

The first observation matches those presented in (Whittaker, 2001) and validates our approach which consists in carefully adapting the ASR and NER models with data corresponding to the exact period of time as the one of the documents to process: by taking into account a larger period of time for the adaptation corpus, the necessity of restraining the models to the most frequent entities would lead to discard low frequency terms that can be crucial for characterizing the news of a given day.

If the second observation clearly highlights the correlation between the NE distribution in both corpora, it also points out that only 25% of the entities of the *unmatched* corpus occur in the newsletters corresponding to the same days. Therefore the potential improvement in the overall NER performance is clearly limited. This will be confirmed in the next section, however one can think that if these entities are shared, for a given day, by both corpora, it is because they represent the key topics of this day and therefore they can be considered as very relevant indexing terms for applications like document retrieval. This last point is developed in section 7.

6.2 Model adaptation

Several studies (Whittaker, 2001; Federico and Bertoldi, 2001; Chen et al., 2004) propose adaptation methods of a general language model to the possibly small corpora extracted from these kinds of metadata information (an overview of these methods can be found in (Bellegarda, 2004)). Depending on the adaptation method and the kind of metadata information used, some gains in performance have been reported. But it appears that the choice of the metadata and the size of the adaptation corpus collected are critical in this adaptation process: if the adaptation corpus is not exactly related to the topics of the document to process, no real gains are obtained (e.g. (Chen et al., 2004) reports that the best gains were obtained with a story-based adaptation method).

From all these previous works, our system implements the following adaptation process:

- the text corpus corresponding to the newsletters is added to the ASR language model by means of a linear interpolation;
- proper names occuring twice or more in the newsletter corpus are added to the ASR lexicon;
- for the same days as those of the *unmatched* corpus, this cutoff is suppressed and all the proper names are added;
- the Named Entity wordlists and grammars are also enriched with these proper names and entities extracted from the collected corpus.

1K new proper names were added to the 65K word ASR lexicon. The general OOV reduction obtained was 0.14% leading to an absolute WER reduction of 0.3%. Similarly the SER decreased of about 0.3% thanks to this adaptation and the Oracle recall measure in the word lattices was improved by an absolute 3%. These improvements are not significant enough to justify the use of this kind of metadata information for improving the general performance of both ASR and NER processes. However, if we focus now on the entities occurring in the newsletters corresponding to the exact days of the *unmatched* corpus, the improvement is much more significant, as presented in the next section.

7 Named Entity Indexation

As previously mentioned, 25% of the *unmatched* corpus entities occur in the newsletters corresponding to the same day as those of the *unmatched* test. In order to measure the improvement obtained with our adaptation technique on these particular entities, we did the following experiment:

- a set of 352 entities was selected from the newsletters related to same period of time as the test, these entities represent the indexing terms that are going to be looked for in the word lattices of the *unmatched* corpus;
- the NER_{asr} system was then applied to these word lattices with two conditions: the word lattices and the NER models before adaptation and those obtained after adaptation with the newsletter corpus;
- precision, recall, F-measure and Oracle error rate were estimated for both conditions.

Condition	Prec.	Recall	F-m	Oracle
no adaptation	87.0	75.7	80.9	83.6
with adaptation	87.5	83.9	85.7	92

Figure 4: Extraction results on the selected NEs on the *unmatched* corpus with and without adaptation of the ASR and NER models on the newsletter corpus

As we can see in table 4, the adaptation process increases very significantly the recall measure of the NE extraction. This is particularly relevant in some IE tasks like the document retrieval task.

8 Conclusion

We have presented in this paper a robust Named Entity Recognition system dedicated to process ASR transcripts. The FSM-based approach allows us to control the generalization capabilities of the system while the statistical tagger provides good labeling decisions. The main feature of this system is its ability to extract n-best lists of NE hypothesis from word lattices leaving the decision strategy choosing to either emphasize the recall or the precision of the extraction, according to the task targeted. A comparison between this approach and a standard approach based on the NLP tools *Lingpipe* validates our hypotheses. This integration of the ASR and the NER processes is particularly important in difficult conditions like those that can be found in large spoken archives where the training corpus does not match all the documents to process. A study of the use of metadata information in order to adapt the ASR and NER models to a specific situation showed that if the overall improvement is small, some salient information related to the metadata added can be better extracted by means of this adaptation.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *ACL'03, Sapporo, Japan*.

D. Appelt and D. Martin. 1999. Named entity extraction from speech: Approach and results using the TextPro system. In *Proceedings Darpa Broadcast News Workshop*.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42 Issue 1:93–108.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. volume 24, pages 211–231.

Andrew Brothwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition.

Langzhou Chen, Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. 2004. Dynamic language modeling for broadcast news. In *In International Conference on Speech and Language Processing*, pages 1281–1284.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

M. Federico and N. Bertoldi. 2001. Broadcast news LM adaptation using contemporary texts. In *Proceedings of European Conference on Speech Communication and Technology (Eurospeech)*, pages 239–242, Aalborg, Denmark.

Jonathan G. Fiscus and George R. Doddington. 2002. Topic detection and tracking evaluation overview. *Topic detection and tracking: event-based information organization*, pages 17–31.

G. Gravier, J.F. Bonastre, E. Geoffrois, S. Galliano, K. McTait, and K. Choukri. 2004. ESTER, une campagne d'évaluation des systèmes d'indexation automatique d'émissions radiophoniques en français. In *Proc. Journées d'Etude sur la Parole (JEP)*.

David Miller, Sean Boisen, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 2000. Named entity extraction from noisy input: Speech and OCR. In *Proceedings of ANLP-NAACL 2000*, pages 316–324.

D. D. Palmer and M. Ostendorf. 2001. Improving information extraction by modeling errors in speech recognizer output. In *Proceedings of the First International Conference on Human Language Technology Research*.

M. A. Przybocki, J. G. Fiscus, J. S. Garofolo, and D. S. Pallett. 1999. 1998 Hub-4 Information Extraction Evaluation. In *Proceedings Of The DARPA Broadcast News Workshop*, pages 13–18. Morgan Kaufmann Publishers.

Bhuvana Ramabhadran, Jing Huang, and Michael Picheny. 2003. Towards automatic transcription of large spoken archives - english ASR for the MALACH project. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 216–219.

Murat Saraclar and Richard Sproat. 2004. Lattice-based search for spoken utterance retrieval. In *HLT-NAACL 2004: Main Proceedings*, pages 129–136, Boston, Massachusetts, USA. Association for Computational Linguistics.

E. W. D. Whittaker. 2001. Temporal adaptation of language models. In *Adaptation Methods for Speech Recognition, ISCA Tutorial and Research Workshop (ITRW)*, August. LM Adaptation for information retrieval of spoken news/radio programs (i.e. SpeechBot).

Lufeng Zhai, Pascale Fung, Richard Schwartz, Marine Carpuat, and Dekai Wu. 2004. Using n-best lists for named entity recognition from chinese speech. In *HLT-NAACL 2004: Short Papers*, pages 37–40, Boston, Massachusetts, USA. Association for Computational Linguistics.

Mining Context Specific Similarity Relationships Using The World Wide Web

Dmitri Roussinov
Department of Information Systems
W.P. Carey School of Business
Arizona State University
Tempe, AZ, 85287
dmitri.roussinov@asu.edu

Leon J. Zhao
Department of Management Information Systems
University of Arizona
Tucson, AZ 85721
lzhao@bpa.arizona.edu

Weiguo Fan
Department of Information Systems
Virginia Tech
Blacksburg, VA 24061
wfan@vt.edu

Abstract

We have studied how context specific web corpus can be automatically created and mined for discovering semantic similarity relationships between terms (words or phrases) from a given collection of documents (*target collection*). These relationships between terms can be used to adjust the standard vectors space representation so as to improve the accuracy of similarity computation between text documents in the target collection. Our experiments with a standard test collection (Reuters) have revealed the reduction of similarity errors by up to 50%, twice as much as the improvement by using other known techniques.

1 Introduction

Many modern information management tasks such as document retrieval, clustering, filtering and summarization rely on algorithms that compute similarity between text documents. For example, clustering algorithms, by definition, place documents similar to each other into the same cluster. Topic detection algorithms attempt to detect documents or passages similar to those already presented to the users. "Query by example" retrieval is based on similarity between a document selected as example and the other ones in the collection. Even a classical retrieval task can be formulated as rank ordering according to the similarity between the document (typically very short) representing user's query and all the documents in the collection.

For similarity computation, text documents are represented by *terms* (words or phrases) that they have, and encoded by vectors according to a predominantly used vector space model (Salton & McGill, 1983). Each coordinate corresponds to a *term* (word or phrase) possibly present within a document. Within that model, a high similarity between a pair of documents can be only indicated by sharing same terms. This approach has apparent limitations due to the notorious vocabulary problem (Furnas et al., 1997): people very often use different words to describe semantically similar objects. For example, within a classical vector space model, the similarity algorithm would treat words *car* and *automobile* as entirely different, ignoring semantic similarity relationship between them.

It has been known for a long time that semantic similarity relationships between terms can be discovered by their co-occurrence in the same documents or in the vicinity of each other within documents (von Rijsbergen, 1977). Until the 1990s, the studies exploring co-occurrence information for building a thesaurus and using it in automated query expansion (adding similar words to the user query) resulted in mixed results (Minker et al., 1972; Peat & Willett, 1991). The earlier difficulties may have resulted from the following reasons:

1) The test collections were small, sometimes only few dozens of documents. Thus, there was only a small amount of data available for statistical co-occurrence analysis (mining), not enough to establish reliable associations.

2) The evaluation experiments were based on retrieval tasks, short, manually composed queries. The queries were at times ambiguous and, as a result, wrong terms were frequently added to the query. E.g. initial query "jaguar" may be expanded with the words "auto", "power", "engine" since they co-occur with "jaguar" in auto related documents. But, if the user was actually referring to an animal then the retrieval accuracy would degrade after the expansion.

3) The expansion models were overly simplistic, e.g. by merely adding more keywords to Boolean queries (e.g. "jaguar OR auto OR power OR car").

Although more recent works removed some of the limitations and produced more encouraging results (Grefenstette, 1994; Church et al., 1991; Hearst et al., 1992; Schutze and Pedersen, 1997; Voorhees, 1994) there are still a number of questions that remain open:

1) What is the range for the magnitude of the improvement. Can the effect be of practical importance?

2) What are the best mining algorithms and formulas? How crucial is the right choice of them?

3) What is the best way to select a corpus for mining? Specifically, is it enough to mine only within the same collection that is involved in retrieval, clustering or other processing (*target collection*), or constructing and mining a larger

external corpus (like a subset of World Wide Web) would be of much greater help?
4) Even if the techniques studied earlier are effective (or not) for query expansion within the document retrieval paradigm, are they also effective for a more general task of document similarity computation? Similarity computation stays behind almost all information retrieval tasks including text document retrieval, summarization, clustering, categorization, query by example etc. Since documents are typically longer than user composed queries, their vector space representations are much richer and thus expanding them may be more reliable due to implicit disambiguation.

Answering these questions constitutes the novelty of our work. We have developed a Context Specific Similarity Expansion (CSSE) technique based on word co-occurrence analysis within pages automatically harvested from the WWW (Web corpus) and performed extensive testing with a well known Reuters collection (Lewis, 1997). To test the similarity computation accuracy, we designed a simple combinatorial metric which reflects how accurately (as compared to human judgments) the algorithm, given a document in the collection, orders all the other documents in the collection by the perceived (computed) similarity. We believe that using this metric is more objective and reliable than trying to include all the traditional metrics specific to each application (e.g. recall/precision for document retrieval, type I/II errors for categorization, clustering accuracy etc.) since the latter may depend on the other algorithmic and implementation details in the system. For example, most clustering algorithms rely on the notion of similarity between text documents, but each algorithm (k-means, minimum variance, single link, etc.) follows its own strategy to maximize similarity within a cluster.

We have found out that our CSSE technique have reduced similarity errors by up to 50%, twice as much as the improvement due to using other known techniques such as Latent Semantic Indexing (LSI) and Pseudo Relevance Feedback (PRF) within the same experimental framework. In addition to this dramatic improvement, we have established the importance of the following for the success of the expansion: 1) using external corpus (a constructed subset of WWW) in addition to the target collection 2) taking the context of the target collection into consideration 3) using the appropriate mining formulas. We suggest that these three crucial components within our technique make it significantly distinct from those explored early and also explain more encouraging results.

The paper is structured as follows. Section 2 discusses previous research results that are closely related to our investigation. Section 3 presents algorithms implemented in our experiments. Section 4 describes our experiments including error reduction, sensitivity analysis, and comparison with other techniques. Finally, Section 5 concludes the paper by explaining our key contributions and outlining our future research.

2 Related Work

Most of the prior works performed only mining within the target collection itself and revealed results ranging from small improvements to negative effects (degrading performance). Throughout our paper, we refer to them as *self-mining* to distinguish from mining *external* corpus, which we believe is more promising for similarity computation between documents due to the following intuitive consideration. Within self-mining paradigm, terms *t1* and *t2* have to frequently co-occur in the collection in order to be detected as associated (synonymic). In that case, expanding document *D* representation with a term *t2* when the document already has term *t1* is not statistically likely to enrich its representation since t2 is likely to be in document *D* anyway. We believe mining external larger and contextually related corpus has the potential to discover more interesting associations with much higher reliability than just from the target collection. That is why, this paper focuses on constructing and mining the external corpus.

There are very few studies that used external corpus and standard evaluation collections. Grefenstette (1994) automatically built a thesaurus and applied it for query expansion, producing better results than using the original queries. Gauch et al. (1998) used one standard collection for mining (TREC4) and another (TREC5) for testing and achieved 7.6% improvement. They also achieved 28.5% improvement on the narrow-domain Cystic Fibrosis collection. Kwok (1998) also reported similar results with TREC non Web collections. Ballesteros and Croft (1998) used unlinked corpora to reduce the ambiguity associated with phrasal and term translation in Cross-Language Retrieval.

There are even fewer studies involving semantic mining on the Web and its methodological evaluation. Géry and Haddad Géry (1999) used about 60,000 documents from one specific domain for mining similarity among French terms and tested the results using 4 ad hoc queries. Sugiura and Etzioni (2000) developed a tool called Q-Pilot that mined the web pages retrieved by commercial search engines and expanded the user query by adding similar terms. They reported preliminary yet encouraging results but tested only the overall system, which includes the other, not directly related to mining features, such as clustering, pseudo-relevance feedback, and selecting the appropriate external search engine. Furthermore, they only used the correctness of the engine selection as the evaluation metric. There are some other well known techniques that do not perform mining for a thesaurus explicitly but still capture and utilize semantic similarity between the terms in an implicit way, namely Latent Semantic Indexing (LSI) and Pseudo Relevance Feedback (PRF). Latent Semantic Indexing (Analysis) (Deerwester et al., 1998) a technique based on Singular Value Decomposition, was studied in a number of works. It reduces the number of dimensions in the document space thus reducing the noise (linguistic variations) and bringing semantically similar terms together, thus it

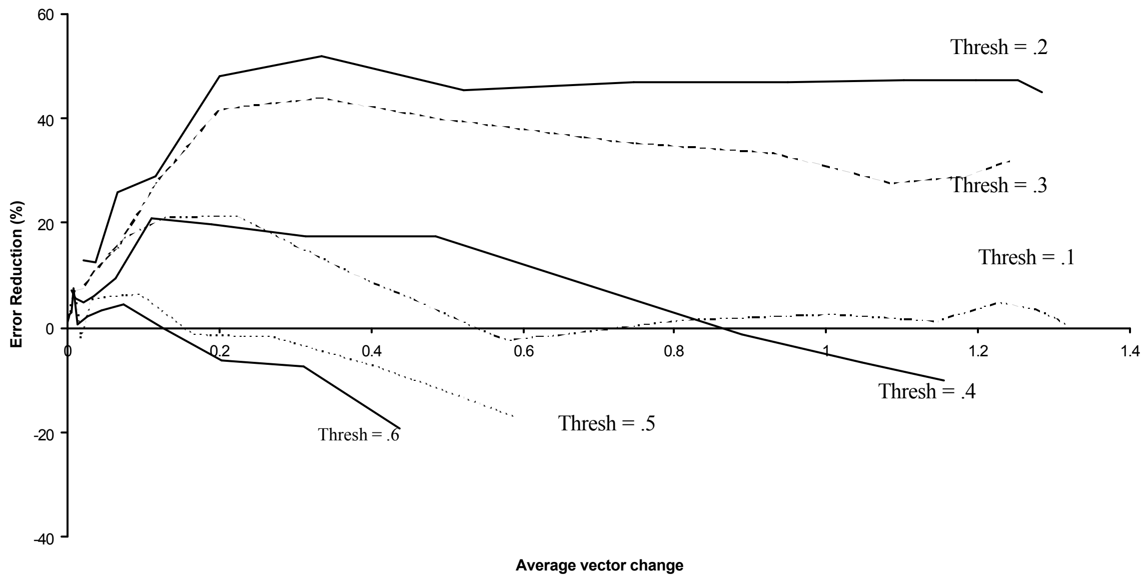


Figure 1. The average error reduction (%) as a function of average document vector change *Ca* for various threshold parameters *Thresh*.

takes into consideration the correlation between the terms. The reported improvements so far however have not exceeded 10-15% in standard collections) and sensitive to the choice of the semantic axis (reduced dimensions). The general idea behind the Pseudo Relevance Feedback (PRF) (Croft & Harper, 1979) or its more recent variation called Local Context Analysis (Xu & Croft, 2000) is to assume that the top rank retrieved documents are relevant and use certain terms from them for the query expansion. A simple approach has been found to increase performance over 23% on the TREC3 and TREC4 collections and became internal part of modern IR systems. Although this idea has been only applied so far to users' queries, we extended it in this study to similarity computation between documents in order to compare with our approach. Although we believe this extension is novel, it is not the focus of this study. It is also worth mentioning that both LSI and PRF fall into "self-mining" category since they do not require external corpus.

A manually built and maintained ontology (a thesaurus), such as WorldNet, may serve as a source of similarity between terms and has been shown to be useful for retrieval tasks (Voorhees, 1994). However, one major drawback of manual approach is high cost of creating and maintaining. Besides, the similarity between terms is context specific. For example, for a campus computer support center the words *student*, *faculty*, *user* are almost synonyms, but for designers of educational software (e.g. Blackboard), the words *student* and *faculty* would represent entirely different roles.

Although the terms "mining", "web mining" and "knowledge discovery" have been used by other researchers in various contexts (Cooley, 1997), we believe it is legitimate to use them to describe our work for two major reasons: 1) We use algorithms and formulas coming from the data mining field, specifically signal to noise ratio association metric (Church, 1989; Church, 1991) 2) Our approach interacts with commercial search engines and harvests web pages contextually close to the target collection, and there is mining of resources (the search engine database) and discovery of content (web pages) involved. We admit that the term "mining" may be also used for a more sophisticated or different kind of processing than our approach here.

3 Algorithms And Implementations

The target collection (Reuters in our experiment) is indexed and its most representative terms are used to construct a corpus from an external source (e. g. World Wide Web). The term-to-term similarity matrix is created by co-occurrence analysis within the corpus and subsequently used to expand document vectors in order to improve the accuracy (correctness) of similarity computation between the documents in the target collection. Although in this work we do not study the effects on the individual applications of the similarity computation, it is crucial for such tasks as retrieval, clustering, categorization or topic detection.

3.1 Building a Web Corpus

We designed and implemented a heuristic algorithm that takes advantage of the capabilities provided by commercial web search engines. In our study, we used AltaVista (www.altavista.com), but most other search engines would also qualify for the task. Ideally, we would like to obtain web pages that

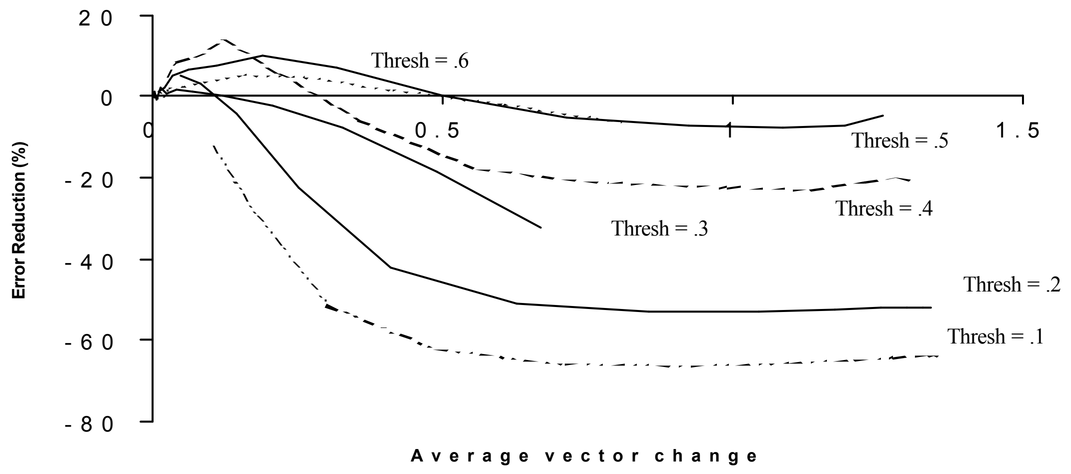


Figure 2. The average error reduction (%) as a function of average document vector change *Ca* for various threshold parameters *Tresh* without "context hint" terms.

contain the terms from the target collection in the similar context. While constructing Web corpus, our spider automatically sends a set of queries to AltaVista and obtains the resulting URLs. The spider creates one query for each term t_i out of 1000 most frequent terms in the target collection (stop words excluded) according to the following formula:

$$q_i = \text{"+"} + t_i + \text{" "} + context\_hint$$

, where + means string concatenation, quotes are used to represent text strings literally and *context_hint* is composed of the top most frequent terms in the target collection (stop words excluded) separated by empty space. Although this way of defining context may seem a bit simplistic, it still worked surprisingly well for our purpose.

According to AltaVista, a word or phrase preceded by '+' sign has to be present in the search results. The presence of the other words and phrases (context hint string in our case) is only desirable but not required. The total number of the context hint terms (108 in this study) is limited by the maximum length of the query string that the search engine can accept.

We chose to use only top 1000 terms for constructing corpus to keep the downloading time manageable. We believe using a larger corpus would demonstrate even larger improvement. Approximately 10% of those terms were phrases. We only used the top 200 hits from each query and only first 20Kbytes of HTML source from each page to convert it into plain text. After removing duplicate URLs and empty pages, we had 19,198 pages in the Web corpus to mine.

Downloading took approximately 6 hours and was performed in parallel, spawning up to 20 java processes at a time, but it still remained the largest scalability bottleneck.

3.2 Semantic Similarity Discovery

CSSE performs co-occurrence analysis at the document level and computes the following values: *df(t1, t2)* is the joint document frequency, i.e., the number of web pages where both terms *t1* and *t2* occur. df(t) is the document frequency of the term t, i.e., the number of web pages in which the term t occurs. Then, CSSE applies a well known signal to noise ratio formula coming from data mining (Church, 1991) to establish similarity between terms *t1* and *t2*:

$$sim(t1, t2) = \log \frac{N \cdot df(t1,t2)}{df(t1) \cdot df(t2)} / \log N, \qquad (1)$$

where *N* is the total number of documents in the mining collection (corpus),

log N is the normalizing factor, so the *sim* value would not exceed *1* and be comparable across collections of different size.

Based on the suggestions from the other studies using formula (1), before running our tests, we decided to discard as spurious all the co-occurrences that happened only within one or two pages and all the similarities that are less than the specified threshold *(Thresh).*

3.3 Vector Expansion

Since we were modifying document vectors (more general case), but not queries as in the majority of prior studies, we refer to the process as *vector expansion*. As we wrote in literature review, there are many possible heuristic ways to perform vector expansion. After preliminary tests, we settled on the simple linear modification with post re-normalization as presented below. The context of the target collection is represented by the similarity matrix *sim(t1, t2)* mined as described in the preceding section. Our vector expansion algorithm adds all the related terms to the vector representation of the document *D* with the weights proportional to the degree of the relationships and the global inverse document frequency (IDF) weighting of the added terms:

$w(t, D)' = w(t, D) +$

$$a \sum_{t' \in d} w(t', D)\, sim(t', t) \log \frac{N}{df(t)}, \text{ where}$$

w(t, D) is the initial, not expanded, weight of the term *t* in the document *D* (assigned according to TF-IDF weighting scheme in our case); *w'(t, D)* is the modified weight of the term *t* in the document *D; t'* iterates through all *(*possibly repeating) terms in the document *D ;* a is the adjustment factor (a parameter controlled in the expansion process).

4 Experiments

4.1 Similarity Error Reduction

Since in this study we were primarily concerned with improving similarity computation but not retrieval per se, we chose a widely used for text categorization Reuters collection (Lewis, 1997) over TREC or similar collections with relevance judgments. We used a modified version of Lewis' (1992) suggestion to derive our evaluation metric, which is similar to the metric derived from Kruskal-Goodman statistics used in Haveliwala et al. (2002) for a study with Yahoo web directory (www.yahoo.com). Intuitively, the metric reflects the probability of algorithm guessing the correct order (called *ground truth*), imposed by a manually created hierarchy (simplified to a partition in Reuters case). Ideally, for each document *D,* the similarity computation algorithm should indicate documents sharing one or more Reuters categories with document D to be more similar to the document *D* than the documents not sharing any categories with *D*. We formalized this intuitive requirement into a metric by the following way. Let's define a test set *Sa* to be the set of all the document triples *(D, D1, D2)* such that $D^{1}D1$, $D^{1}D2$, $D1^{1}D2$, and furthermore *D* shares at least one common category with *D1* but no common categories with *D2*. We defined *total error count* (*Ec*) as the number of triples in the test set *Sa* such that *sim(D, D1) < sim(D, D2)* since it should be the other way around. Our accuracy metric reported below is the total error count normalized by the size of the test set *Sa*: *similarity error = Ec / #Sa,* computed for each Reuters topics and averaged across all of them. The metric ranges from 0 (ideal case) to .5 (random ordering). It also needed an adjustment to provide the necessary continuity as justified in the following. Since the documents are represented by very sparse vectors, very often (about 5% of all triples) documents *D, D1, D2* do not have any terms in common and as a result similarity computation results in a tie: *sim(D,D1) = sim (D, D2)*. A tie can not be considered an error because in that case one can suggest a trivial improvement to the similarity algorithm by simply breaking the ties at random in any direction with an equal chance, and thus reducing errors in 50% of all ties. This is why the metric counts half of all the ties as errors, which completely removes this discontinuity.

We used all the Reuters *78* topics from the "commodity code" group since they are the most "semantic", not trying the others (Economic Indicator Codes, Currency Codes, Corporate Codes). We discarded the topics that had only 1 document and used only the documents that had at least one of the topics. This reduced our test collection to *1841* documents, still statistically powerful and computationally demanding since millions of triples had to be considered (even after some straightforward algorithmic optimizations). After indexing and stemming (Porter, 1980) the total number of unique stems used for the vector representation was *11461*.

Weighting Scheme	boolean vectors	TF only	IDF only	TF-IDF
Similarity Error	0.1750	0.1609	0.1278	0.1041

Table 2. Comparison of different weighting schemes with the original (not expanded) documents.

Table 2 lists the similarity error averaged by topics for the different weighting schemes we tried first in our experiment. Since TF-IDF weighting was by far the best in this evaluation set up, we limited our expansion experiments to TF-IDF scheme only. For similarity measure between document vectors, we used the most common negative Euclidian distance after normalizing the vectors to unit length. It can be shown, that cosine metric (dot product), the other popular metric, results in the same order and, thus same similarity error as well. Without normalization or stemming the errors were almost twice as much larger.

Although we varied the adjustment parameter *a* in our experiment, for better interpretation, we plotted our primary metric (average error reduction) as a function of *Ca*, the average Euclidian distance between the original and the modified document vectors when both vectors are normalized to unit length. *Ca* serves as a convenient parameter controlling the degree of change in the document vectors, better than *a*, because same values of *a* may result in different changes depending on the term-to-term similarity matrix *sim(t1, t2)*. In theory, *Ca* varies from *0* (no change) to $\sqrt{2}$, the case of maximum possible change (no common terms between initial and expanded representation). By varying adjustment factor *a* from 0 to 10 and higher we observed almost the entire theoretical range of *Ca*: starting from negligible change and going all the way to $\sqrt{2}$, where the added terms entirely dominated the original ones. The average number of terms in the document representation was in 60-70 range before expansion and in 200-300 range after the expansion. This of course increased computational burden. Nevertheless, even after the expansion, the vector representations still remained sparse and we were able to design and implement some straightforward algorithmic improvements taking advantage of this sparsity to keep processing time manageable. The expansion for entire Reuters collection was taking less than one minute on a workstation with Pentium III 697 MHz processor, 256 MB of RAM, with all the sparse representations of the documents and similarity matrix stored in primary memory. This renders the expansion suitable for online processing.

To evaluate the performance of each technique, we used the error reduction (%) relatively to the baseline shown in Table 1 (TF-IDF column) averaged across all the topics, which corresponds to the lowest

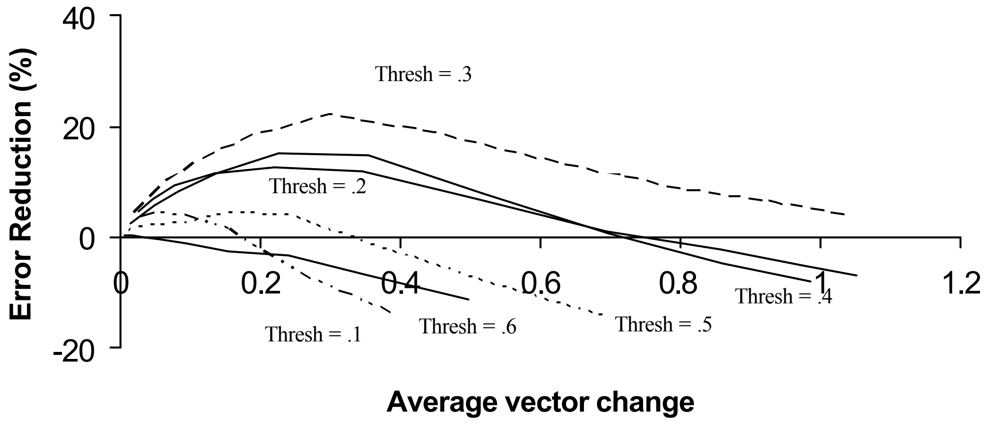


Figure 3. The average error reduction (%) as a function of average document vector change *Ca* for various threshold parameters *Tresh* without using external mining collection.

original non-expanded similarity error. Figure 1 shows the error reduction as a function of *Ca* for various values of *Thresh*. We stopped increasing *Ca* once the improvement dropped below -10% to save testing time. Several facts can be observed from the results:

1) The error reduction for *Thresh* in the mid range of *Ca [.2-.4]* is very stable, achieves *50%* , which is very large compared with the other known techniques we used for comparison as discussed below. The effect is also comparable with the difference between various weighting functions (Table 2), which we believe renders the improvement practically significant.

2) For small thresholds (*Thresh* < *.1*), the effect is not that stable, possibly since many non-reliable associations are involved in the expansion.

3) Larger thresholds (*Thresh* > *.4*) are also not very reliable since they result in a small number of associations created, and thus require large values of adjustment parameter *a* in order to produce substantial average changes in the document vectors (*Ca*), which results in too drastic change in some document vectors.

4) The error reduction curve is unimodal: it starts from 0 for small *Ca*, since document vectors almost do not change, and grows to achieve maximum for *Ca* somewhere in relatively wide *.1 - .5* range. Then, it decreases, because document vectors may be drifting too far from the original ones, falling below *0* for some large values of *Ca*.

5) For thresholds (*Thresh*) *.2* and *.3*, the effect stays positive even for large values of *Ca*, which is an interesting phenomenon because document vectors are getting almost entirely replaced by their expanded representations.

Some sensitivity of the results with respect to the parameters *Thresh, Ca* is a limitation as occurs similarly to virtually all modern IR improvement techniques. Indeed, Latent Semantic Indexing (LSI) needs to have number of semantic axis to be correctly set, otherwise the performance may degrade. Pseudo Relevance Feedback (PRF) depends on several parameters such as number of documents to use for feedback, adjustment factor, etc. All previously studied expansion techniques depend on the adjustment factor as well. The specific choice of the parameters for real life applications is typically performed manually based on trial and error or by following a machine learning approach: splitting data into training and testing sets. Based on the above results, the similarity threshold (*Thresh*) in *.2-.4* and *Ca* in *.1-.5* range seem to be a safe combination, not degrading and likely to significantly (*20-50%*) improve performance. The performance curve being unimodal with respect to both *Ca* and *Thresh* also makes it easier to tune by looking for maxima. Although we have involved only one test collection in this study, this collection (Reuters) varies greatly in the content and the size of the documents, so we hope our results will generalize to other collections.

We also verified that the effect typically diminishes when the size of the mining collection (corpus) is reduced by random sub-sampling. Those results were also similar to those obtained 4 months earlier, although only 80% of the pages in the mining corpus remained.

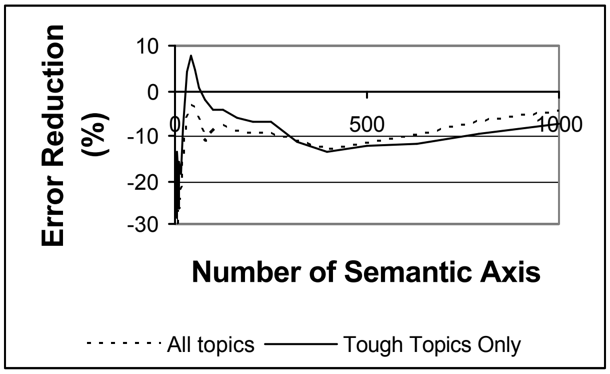


Figure 4. Comparing to LSI.

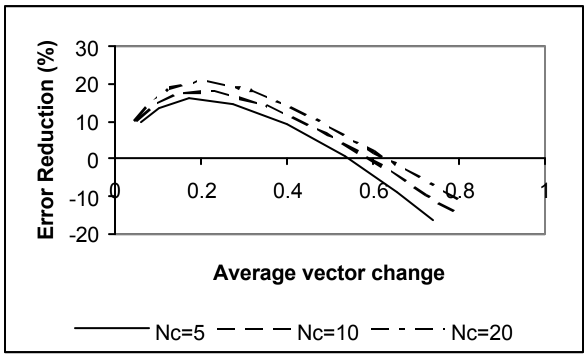


Figure 5. The error reduction as the function of the average vector change due to Pseudo Relevance Feedback for several cut-off numbers Nc.

4.2 Sensitivity Analysis

To test the importance of the context, we removed the "context hint" terms from the queries used by our agent, and created another (less context specific) corpus for mining. We obtained *175,336* unique URLs, much more than with using "context hint" terms since the overlap between different query results was much smaller. We randomly selected *25,000 URLs* of them and downloaded the referred pages. Then, to make the comparison more objective, we randomly selected *19,198* pages (same number as with using context hint) of the non-empty downloaded pages. We mined the similarity relationships from the selected documents in the same way as described above. The resulting improvement (shown in the Figure 2) was indeed much smaller (*13%* and less) than with using "context hint" terms. It also degrades much quicker for larger *Ca* and more sensitive to the choice of *Thresh*. This may explain why mixed results were reported in the literate when the similarity thesaurus was constructed in a very general setting, but not specifically for the target collection in mind. It is also interesting to note a similar behavior of error reduction as the function of *Ca* and *Thresh*: it is unimodal with maximum in approximately same range of arguments. This may also serve as indirect evidence of stability of the effect (even if smaller in that case) with respect to the parameters involved.

To verify the importance of using external corpus vs. self-mining, we mined the similarity relationships from the same collection (Reuters) that we used for the tests (target collection) using the same mining algorithms. Figure 3 shows that the effect of such "self-mining" is relatively modest (up to 20%), confirming that using the external corpus (the Web in our approach) was crucial. Again, the behavior of the error reduction (even smaller in that case) with respect to *Ca* and *Thresh* is similar to the context specific web corpus mining.

4.3 Comparison with Other Techniques

Figure 4 shows the similarity error reduction as a function of the number of semantic axis when LSI is applied. The effect with the entire collection (second column) is always negative. So, the Reuters collection in our experiment set up was found to be not a good application of LSI technique, possibly because many of the topics have already small errors even before applying LSI. To verify our implementation and the applicability of LSI to the similarity computation, we applied it only to the "tougher" 26 topics, those in the upper half if ordered by the original similarity error. As Figure 4 reveals, LSI is effective in that case for numbers of semantic axis comparable with number of topics in the target collection. Our findings are well in line with reported in prior research.

We adapted the classic Pseudo Relevance Feedback algorithm (Qiu, 1993), which has been so far applied only to document retrieval tasks, to similarity computation in a straightforward way and also tried several variations of if (not described here due to lack of space). Figure 5 shows the effect as a function of adjustment factor *a* for various cut-off parameters *Nc* (the number of top ranked documents used for feedback). The effect achieves the maximum of around *21%*, consistent with the results reported in prior research. The improvement is close in magnitude to the one due to "self-mining" described above. We do not claim that our approach is better than PRF since it is not entirely meaningful to make this comparison due to the number of parameters and implementation details involved in both. Also, more important, the techniques rely on different source of data: PRF is a "self-mining" approach while CSSE builds and mines external corpus. Thus, CSSE can be used in addition to PRF.

5 Conclusions

In this paper, we proposed and empirically studied an approach to improve similarity computation between text documents by creating a context specific Web corpus and performing similarity mining within it. The results demonstrated that the similarity errors can be reduced by additional 50% after all the standard procedures such as stemming, term weighting, and vector normalization. We also established the crucial importance of the following three factors, which we believe make our technique distinct from those already explored early and explain more encouraging results that we obtained: *1) Using external corpus. 2) Taking the context of the target collection into consideration. 3) Using the appropriate mining formula.* Another important distinction and possible explanation of a more dramatic effect is our focus on similarity computation between text documents, rather than on document retrieval tasks, which have been more extensively studied in the past. Similarity computation is a more general procedure, which in turns defines the quality of virtually all other specific tasks such as document retrieval, summarization, clustering, categorization, topic detection, query by example, etc. Our future plans are to overcome some of the limitations in this study, specifically using more than a single (although standard and very diverse) collection and study other experimental setups, such as document retrieval, text categorization, or topic detection and tracking.

6 Acknowledgement

Weiguo Fan's work is supported by NSF under the grant number ITR0325579.

References

Church, K.W., Gale, W., Hanks, P., Hindle, D. (1991). Using Statistics in Lexical Analysis. In: Uri Zernik (ed.), Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon. New Jersey: Lawrence Erlbaum, 1991, pp. 115-164.

Church, K.W., Hanks, P. (1989). Word Association Norms, Mutual Information and Lexicography. *In Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, 1989, pp. 76-83.

Cooley, R., Mobasher, B. and Srivastava, J. (1997). Web Mining: Information and Pattern Discovery on the World Wide Web (with R. Cooley and J. Srivastava), in *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

Croft, W.B., and Harper, D.J. (1979). Using probabilistic models of document retrieval without relevance information. Journal of Documentation, 35, pp. 285-295.

Deerwester S., Dumais S., Furnas G., Landauer T.K., and Harshman R., Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41 (1990), 391-407.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The Vocabulary Problem in Human-System Communication. *Communications of the ACM*, 30(11), pp. 964-971.

Géry, M., Haddad, M. H. (1999). Knowledge Discovery for Automatic Query Expansion on the World Wide Web. International Workshop on the World-Wide Web and Conceptual Modeling (WWWCM'99), in conjunction with the 18th International Conference on Conceptual Modeling (ER'99), Paris, France, November 15-18, 1999, pp. 334-347.

Grefenstette, G. (1994). Explorations in Automatic Thesaurus Discovery. Kluwer Academic Publishers, Moston, MA.

Haveliwala, T.H, Gionis, A., Klein, D., Indyk, P. (2002). Evaluating Strategies for Similarity Search on the Web. WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA.

Hearst, M. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora, *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France, July 1992.

Kwok, K.L. (1998). Improving two-stage ad-hoc retrieval for short queries. *Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 250-256, New York, August 1998.

Lewis, D.D. (1992). Representation and Learning in Information Retrieval. *Doctoral Dissertation.* University of Massachusetts at Amherst.

Lewis, D.D. (1997). Reuters-21578 text categorization test collection, Distribution 1.0, Sept 26, 1997.

Minker, J., Wilson, G. A. & Zimmerman, B. H. (1972). An evaluation of query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval*, pp. 329-348.

Peat, H. J. & Willett, P. (1991). The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5), pp. 378-383.

Porter, M.F. (1980). An algorithm for suffix stripping. *Program*, 14, pp. 130--137, 1980.

Qiu, Y. (1993). Concept Based Query Expansion. Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval.

Salton, G. and McGill, M.J. (1983). *Introduction to Modern Information Retrieval.* New York. McGraw-Hill.

Schutze, H. and Pedersen, J.O. (1997). A co-occurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management.* 33(3), pp. 307-318.

Sugiura, A., and Etzioni, O. (2000). Query Routing for Web Search Engines: Architecture and Experiments. *9th International World Wide Web Conference*, Amsterdam, May 15-19, 2000.

van Rijsbergen, C.J.. (1977). A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106--119, 1977.

Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. *In Proceedings of the 17th Annual International ACM/SIGIR Conference*, pp. 61-69, Dublin, Ireland.

Xu, J. and Croft, W.B. (2000). Improving the effectiveness of information retrieval with local context analysis. ACM Transactions on Information Systems (TOIS), 18(1):79--112, 2000.

Ballesteros, L., Croft, W.B. (1998). Resolving Ambiguity for Cross-Language Retrieval. *In Proceedings of the 21th Annual International ACM/SIGIR Conference*, pp. 64-71.

Hidden–Variable Models for Discriminative Reranking

Terry Koo
MIT CSAIL
maestro@mit.edu

Michael Collins
MIT CSAIL
mcollins@csail.mit.edu

Abstract

We describe a new method for the representation of NLP structures within reranking approaches. We make use of a conditional log–linear model, with hidden variables representing the assignment of lexical items to word clusters or word senses. The model learns to automatically make these assignments based on a discriminative training criterion. Training and decoding with the model requires summing over an exponential number of hidden–variable assignments: the required summations can be computed efficiently and exactly using dynamic programming. As a case study, we apply the model to parse reranking. The model gives an F–measure improvement of $\approx 1.25\%$ beyond the base parser, and an $\approx 0.25\%$ improvement beyond the Collins (2000) reranker. Although our experiments are focused on parsing, the techniques described generalize naturally to NLP structures other than parse trees.

1 Introduction

A number of recent approaches in statistical NLP have focused on *reranking* algorithms. In reranking methods, a baseline model is used to generate a set of candidate output structures for each input in training or test data. A second model, which typically makes use of more complex features than the baseline model, is then used to rerank the candidates proposed by the baseline. Reranking approaches have given improvements in accuracy on a number of NLP problems including parsing (Collins, 2000; Charniak and Johnson, 2005), machine translation (Och and Ney, 2002; Shen et al., 2004), information extraction (Collins, 2002), and natural language generation (Walker et al., 2001).

The success of reranking approaches depends critically on the choice of *representation* used by the reranking model. Typically, each candidate structure (e.g., each parse tree in the case of parsing) is mapped to a feature–vector representation. Previous work has generally relied on two approaches to representation: explicitly hand–crafted features (e.g., in Charniak and Johnson (2005)) or features defined through kernels (e.g., see Collins and Duffy (2002)).

This paper describes a new method for the representation of NLP structures within reranking approaches. We build on the intuition that lexical items in natural language often fall into word clusters (for example, *president* and *chairman* might belong to the same cluster) or fall into distinct word senses (e.g., *bank* might have two distinct senses). Our method involves a hidden–variable model, where the hidden variables correspond to an assignment of words to either clusters or word–senses. Lexical items are automatically assigned their hidden values using unsupervised learning within a discriminative reranking approach.

We make use of a conditional log–linear model for our task. Formally, hidden variables within the log–linear model consist of *global* assignments, where a global assignment entails an assignment of every word in the sentence to some hidden cluster or sense value. The number of such global assignments grows exponentially fast with the length of the sentence being processed. Training and decoding with the model requires summing over the exponential number of possible global assignments, a major technical challenge in our model. We show that the required summations can be computed efficiently and exactly using dynamic–programming methods (i.e., the belief propagation algorithm for Markov random fields (Yedidia et al., 2003)) under certain restrictions on features in the model.

Previous work on reranking has made heavy use of lexical statistics, but has treated lexical items as atoms. The motivation for our method comes from the observation that statistics based on lexical items are critical, but that these statistics suffer considerably from problems of data sparsity and word–

sense polysemy. Our model has the ability to alleviate data sparsity issues by learning to assign words to word clusters, and can mitigate problems with word–sense polysemy by learning to assign lexical items to underlying word senses based upon contextual information. A critical difference between our method and previous work on unsupervised approaches to word–clustering or word–sense discovery is that our model is trained using a discriminative criterion, where the assignment of words to clusters or senses is driven by the reranking task in question.

As a case study, in this paper we focus on syntactic parse reranking. We describe three model types that can be captured by our approach. The first method emulates a clustering operation, where the aim is to place similar words (e.g., *president* and *chairman*) into the same cluster. The second method emulates a *refinement* operation, where the aim is to recover distinct senses underlying a single word (for example, distinct senses underlying the noun *bank*). The third definition makes use of an existing ontology (i.e., WordNet (Miller et al., 1993)). In this case the set of possible hidden values for each word corresponds to possible WordNet senses for the word.

In experimental results on the Penn Wall Street Journal treebank parsing domain, the hidden–variable model gives an F–measure improvement of $\approx 1.25\%$ beyond a baseline model (the parser described in Collins (1999)), and gives an $\approx 0.25\%$ improvement beyond the reranking approach described in Collins (2000). Although the experiments in this paper are focused on parsing, the techniques we describe generalize naturally to other NLP structures such as strings or labeled sequences. We discuss this point further in Section 6.1.

2 Related Work

Various machine–learning methods have been used within reranking tasks, including conditional log–linear models (Ratnaparkhi et al., 1994; Johnson et al., 1999), boosting methods (Collins, 2000), variants of the perceptron algorithm (Collins, 2002; Shen et al., 2004), and generalizations of support–vector machines (Shen and Joshi, 2003). There have been several previous approaches to parsing using log–linear models and hidden variables. Riezler et al. (2002) describe a discriminative LFG parsing model that is trained on standard (syntax only) treebank annotations by treating each tree as a full LFG analysis with an observed c-structure and hidden f-structure. Clark and Curran (2004) present an alternative CCG parsing approach that divides each CCG parse into a dependency structure (observed) and a derivation (hidden). More recently, Matsuzaki et al. (2005) introduce a probabilistic CFG augmented with hidden information at each nonterminal, which gives their model the ability to tailor itself to the task at hand. The form of our model is closely related to that of Quattoni et al. (2005), who describe a hidden–variable model for object recognition in computer vision.

The approaches of Riezler et al., Clark and Curran, and Matsuzaki et al. are similar to our own work in that the hidden variables are exponential in number and must be handled with dynamic–programming techniques. However, they differ from our approach in the definition of the hidden variables (the Matsuzaki et al. model is the most similar). In addition, these three approaches don't use reranking, so their features must be restricted to local scope in order to allow dynamic–programming approaches to training. Finally, these approaches use Viterbi or other approximations during decoding, something our model can avoid (see section 6.2).

In some instantiations, our model effectively clusters words into categories. Our approach differs from standard word clustering in that the clustering criteria is directly linked to the reranking objective, whereas previous word–clustering approaches (e.g. Brown et al. (1992) or Pereira et al. (1993)) have typically leveraged distributional similarity. In other instantiations, our model establishes word–sense distinctions. Bikel (2000) has done previous work on incorporating the WordNet hierarchy into a generative parsing model; however, this approach requires data with word–sense annotations whereas our model deals with word–sense ambiguity through unsupervised discriminative training.

3 The Hidden–Variable Model

In this section we describe a hidden–variable model based on conditional log–linear models. Each sentence s_i for $i = 1 \ldots n$ in our training data has a set of n_i candidate parse trees $t_{i,1}, \ldots, t_{i,n_i}$, which are the output of an N–best baseline parser. Each candidate parse has an associated F–measure score, indicating its similarity to the gold–standard parse. Without loss of generality, we define $t_{i,1}$ to be the parse with the highest F–measure for sentence s_i.

Given a candidate parse tree $t_{i,j}$, the hidden–variable model assigns a domain of hidden values to each word in the tree. For example, the hidden–value domain for the word *bank* could be $\{bank_1, bank_2, bank_3\}$ or $\{\mathsf{NN}_1, \mathsf{NN}_2, \mathsf{NN}_3\}$. Detailed descriptions of the domains we used are given in Section 4.1. Formally, if $t_{i,j}$ spans m words then the hidden–value domains for each word are the sets $H_1(t_{i,j}), \ldots, H_m(t_{i,j})$. A global hidden–value assignment, which attaches a hidden value to every word in $t_{i,j}$, is written $\mathbf{h} = (h_1, \ldots, h_m) \in \mathbf{H}(t_{i,j})$, where $\mathbf{H}(t_{i,j}) = H_1(t_{i,j}) \times \ldots \times H_m(t_{i,j})$ is the set of all possible global assignments for $t_{i,j}$.

We define a feature–based representation Φ such that $\Phi(t_{i,j}, \mathbf{h}) \in \mathbb{R}^d$ is a vector of feature occurrence counts that describes candidate parse $t_{i,j}$ with global assignment $\mathbf{h} \in \mathbf{H}(t_{i,j})$. We write Φ_k for $k = 1 \ldots d$ to denote the k^{th} component of the vector Φ. Each component of the feature vector is the count of some substructure within $(t_{i,j}, \mathbf{h})$. For example, Φ_{12} and Φ_{101} could be defined as follows:

$$\begin{aligned} \Phi_{12}(t_{i,j}, \mathbf{h}) &= \text{Number of times the word } \textit{the} \text{ occurs with hidden value } \textit{the}_3 \text{ and part of speech tag } \mathsf{DT} \text{ in } (t_{i,j}, \mathbf{h}). \\ \Phi_{101}(t_{i,j}, \mathbf{h}) &= \text{Number of times } \textit{CEO}_1 \text{ appears as the subject of } \textit{owns}_2 \text{ in } (t_{i,j}, \mathbf{h}) \end{aligned} \quad (1)$$

We use a parameter vector $\Theta \in \mathbb{R}^d$ to define a log–linear distribution over candidate trees together with global hidden–value assignments:

$$p(t_{i,j}, \mathbf{h} \mid s_i, \Theta) = \frac{e^{\Phi(t_{i,j},\mathbf{h})\cdot\Theta}}{\sum_{j',\mathbf{h}' \in \mathbf{H}(t_{i,j'})} e^{\Phi(t_{i,j'},\mathbf{h}')\cdot\Theta}}$$

By marginalizing out the global assignments, we obtain a distribution over the candidate parses alone:

$$p(t_{i,j} \mid s_i, \Theta) = \sum_{\mathbf{h} \in \mathbf{H}(t_{i,j})} p(t_{i,j}, \mathbf{h} \mid s_i, \Theta) \quad (2)$$

Later in this paper we will describe how to train the parameters of the model by minimizing the following loss function—which is the negative log–likelihood of the training data—with respect to Θ:

$$\begin{aligned} L(\Theta) &= -\sum_i \log p(t_{i,1} \mid s_i, \Theta) \\ &= -\sum_i \log \textstyle\sum_{\mathbf{h} \in \mathbf{H}(t_{i,1})} p(t_{i,1}, \mathbf{h} \mid s_i, \Theta) \end{aligned} \quad (3)$$

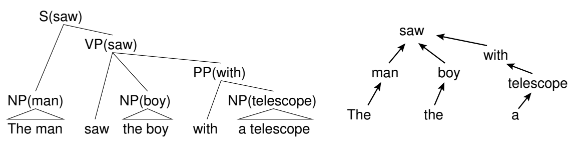


Figure 1: A sample parse tree and its dependency tree.

3.1 Local Feature Vectors

Note that the number of possible global assignments (i.e., $|\mathbf{H}(t_{i,j})|$) grows exponentially fast with respect to the number of words spanned by $t_{i,j}$. This poses a problem when training the model, or when calculating the probability of a parse tree through Eq. 2. This section describes how to address this difficulty by restricting features to sufficiently local scope. In Section 3.2 we show that this restriction allows efficient training and decoding of the model.

The restriction to local feature–vectors makes use of the dependency structure underlying the parse tree $t_{i,j}$. Formally, for tree $t_{i,j}$, we define the corresponding dependency tree $\mathbf{D}(t_{i,j})$ to be a set of edges between words in $t_{i,j}$, where $(u, v) \in \mathbf{D}(t_{i,j})$ if and only if there is a head–modifier dependency between words u and v. See Figure 1 for an example dependency tree. We restrict the definition of Φ in the following way[1]. If w, u and v are word indices, we introduce single–variable local feature vectors $\phi(t_{i,j}, w, h_w) \in \mathbb{R}^d$ and pairwise local feature vectors $\phi(t_{i,j}, u, v, h_u, h_v) \in \mathbb{R}^d$. The global feature vector $\Phi(t_{i,j}, \mathbf{h})$ is then decomposed into a sum over the local feature vectors:

$$\begin{aligned} \Phi(t_{i,j}, \mathbf{h}) = &\sum_{1 \le w \le m} \phi(t_{i,j}, w, h_w) + \\ &\sum_{(u,v) \in \mathbf{D}(t_{i,j})} \phi(t_{i,j}, u, v, h_u, h_v) \end{aligned} \quad (4)$$

Notice that the second sum, over pairwise local feature vectors, respects the dependency structure $\mathbf{D}(t_{i,j})$. Section 3.2 describes how this decomposition of Φ leads to an efficient and exact dynamic–programming approach that, during training, allows us to calculate the gradient $\frac{\partial L}{\partial \Theta}$ and, during testing, allows us to find the most probable candidate parse.

In our implementation, each dimension of the local feature vectors is an indicator function signaling the presence of a feature, so that a sum over local feature vectors in a tree gives the occurrence count

[1]Note that the restriction on local feature vectors only concerns the inclusion of hidden values; features may still observe arbitrary structure within the underlying parse tree $t_{i,j}$.

of features in that tree. For instance, define

$$\phi_{12}(t_{i,j}, w, h_w) = \left[\!\left[\begin{array}{l} h_w = \textit{the}_3 \text{ and tree } t_{i,j} \text{ assigns word} \\ w \text{ to part–of–speech DT} \end{array} \right]\!\right]$$

$$\phi_{101}(t_{i,j}, u, v, h_u, h_v) = \left[\!\left[\begin{array}{l} (h_u, h_v) = (\textit{CEO}_1, \textit{owns}_2) \\ \text{and tree } t_{i,j} \text{ places } (u, v) \text{ in} \\ \text{a subject–verb relationship} \end{array} \right]\!\right]$$

where the notation $[\![\mathcal{P}]\!]$ signifies a $0/1$ indicator of predicate $\mathcal{P}$. When summed over the tree, these definitions of ϕ_{12} and ϕ_{101} yield global features Φ_{12} and Φ_{101} as given in the previous example (see Eq. 1).

3.2 Training the Model

We now describe how the loss function in Eq. 3 can be optimized using gradient descent. The gradient of the loss function is given by:

$$\frac{\partial L}{\partial \Theta} = -\sum_i F(t_{i,1}, \Theta) + \sum_{i,j} p(t_{i,j} \,|\, s_i, \Theta) F(t_{i,j}, \Theta)$$

where $F(t_{i,j}, \Theta) = \sum_{\mathbf{h} \in \mathbf{H}(t_{i,j})} \frac{p(t_{i,j}, \mathbf{h} \,|\, s_i, \Theta)}{p(t_{i,j} \,|\, s_i, \Theta)} \Phi(t_{i,j}, \mathbf{h})$ is the expected value of the feature vector produced by parse tree $t_{i,j}$. As we remarked earlier, $|\mathbf{H}(t_{i,j})|$ is exponential in size so direct calculation of either $p(t_{i,j} \,|\, s_i, \Theta)$ or $F(t_{i,j}, \Theta)$ is impractical. However, using the feature–vector decomposition in Eq. 4, we can rewrite the key functions of Θ as follows:

$$p(t_{i,j} \,|\, s_i, \Theta) = \frac{Z_{i,j}}{\sum_{j'} Z_{i,j'}}$$

$$F(t_{i,j}, \Theta) = \sum_{\substack{1 \le w \le m \\ h_w \in H_w(t_{i,j})}} p(t_{i,j}, w, h_w) \phi(t_{i,j}, w, h_w) + \sum_{\substack{(u,v) \in \mathbf{D}(t_{i,j}) \\ h_u \in H_u(t_{i,j}) \\ h_v \in H_v(t_{i,j})}} p(t_{i,j}, u, v, h_u, h_v) \phi(t_{i,j}, u, v, h_u, h_v)$$

where $p(t_{i,j}, w, h_w)$ and $p(t_{i,j}, u, v, h_u, h_v)$ are marginalized probabilities and $Z_{i,j}$ is the associated normalization constant:

$$Z_{i,j} = \sum_{\mathbf{h} \in \mathbf{H}(t_{i,j})} e^{\Phi(t_{i,j}, \mathbf{h}) \cdot \Theta}$$

$$p(t_{i,j}, w, x) = \sum_{\mathbf{h} \,|\, h_w = x} p(t_{i,j}, \mathbf{h} \,|\, s_i, \Theta)$$

$$p(t_{i,j}, u, v, x, y) = \sum_{\mathbf{h} \,|\, h_u = x, h_v = y} p(t_{i,j}, \mathbf{h} \,|\, s_i, \Theta)$$

The three quantities above can be computed with belief propagation (Yedidia et al., 2003), a dynamic–programming technique that is efficient[2] and exact when the graph $\mathbf{D}(t_{i,j})$ is a tree, which is the case in our parse reranking model. Having calculated the gradient in this way, we minimize the loss using stochastic gradient descent[3] (LeCun et al., 1998).

4 Features for Parse Reranking

The previous section described hidden–variable models for discriminative reranking. We now describe features for the parse reranking problem. We focus on the definition of hidden–value domains and local feature vectors in the reranking model.

4.1 Hidden–Value Domains and Local Features

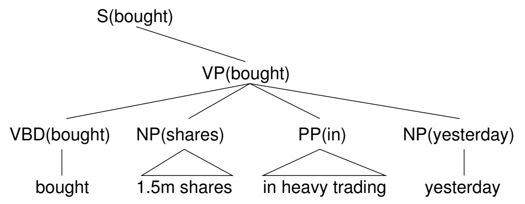
Each word in a parse tree is given a domain of possible hidden values by the hidden–variable model. Models with widely varying behavior can be created by changing the way these domains are defined. In particular, in this section we will see how different definitions of the domains give rise to the three main model types: clustering, refinement, and mapping into a pre–built ontology such as WordNet.

As illustration, consider a simple approach that splits each word into a domain of three word–sense hidden values (e.g., the word *bank* would yield the domain $\{\textit{bank}_1, \textit{bank}_2, \textit{bank}_3\}$). In this approach, each word receives a domain of hidden values that is not shared with any other word. The model is then able to distinguish several different usages for each word, emulating a refinement operation. An alternative approach is to split each word's part–of–speech tag into several sub–tags (e.g., *bank* would yield $\{\text{NN}_1, \text{NN}_2, \text{NN}_3\}$). This approach assigns the same domain to many words; for instance, singular nouns such as *bond*, *market*, and *bank* all receive the same domain. The behavior of the model then emulates a clustering operation.

Figure 2 shows the single–variable and pairwise features used in our experiments. The features are shown with hidden variables corresponding to word–specific hidden values, such as $\textit{shares}_1$ or $\textit{bought}_3$. In our experiments, we made use of features such as those in Figure 2 in combination with the following four definitions of the hidden–value

[2]The running time of belief propagation varies linearly with the number of nodes in $\mathbf{D}(t_{i,j})$ and quadratically with the cardinality of the largest hidden–value domain.

[3]We also performed some experiments using the conjugate gradient descent algorithm (Johnson et al., 1999). However, we did not find a significant difference between the performance of either method. Since stochastic gradient descent was faster and required less memory, our final experiments used the stochastic gradient method.



Single–variable features generated for $(shares_1) =$

$(shares_1)$
$(shares_1,\ \text{Word: } shares)$
$(shares_1,\ \text{POS: NN})$
$(shares_1,\ \text{Word: } shares,\ \text{POS: NN})$
$(shares_1,\ \text{Highest NT: NP})$
$(shares_1,\ \text{Word: } shares,\ \text{Highest NT: NP})$
$(shares_1,\ \text{POS: NN},\ \text{Highest NT: NP})$
$(shares_1,\ \text{Word: } shares,\ \text{POS: NN},\ \text{Highest NT: NP})$
$(shares_1,\ \text{Up Path: NP, VP, S})$
$(shares_1,\ \text{Word: } shares,\ \text{Up Path: NP, VP, S})$
$(shares_1,\ \text{Down Path: NP, NN})$
$(shares_1,\ \text{Word: } shares,\ \text{Down Path: NP, NN})$
$(shares_1,\ \text{Head Rule: NP} \rightarrow \text{CD, CD, NNS}_{\text{head}})$
$(shares_1,\ \text{Word: } shares,\ \text{Head Rule: NP} \rightarrow \text{CD, CD, NNS}_{\text{head}})$
$(shares_1,\ \text{Mod Rule: VP} \rightarrow \text{VBD}_{\text{head}}, \text{NP}_{\text{mod}}, \text{PP, NP})$
$(shares_1,\ \text{Word: } shares,\ \text{Mod Rule: VP} \rightarrow \text{VBD}_{\text{head}}, \text{NP}_{\text{mod}}, \text{PP, NP})$
$(shares_1,\ \text{Head Gpar Rule: VP} \rightarrow \text{NP} \rightarrow \text{CD, CD, NNS}_{\text{head}})$
$(shares_1,\ \text{Word: } shares,\ \text{Head Gpar Rule: VP} \rightarrow \text{NP} \rightarrow \text{CD, CD, NNS}_{\text{head}})$
$(shares_1,\ \text{Mod Gpar Rule: S} \rightarrow \text{VP} \rightarrow \text{VBD}_{\text{head}}, \text{NP}_{\text{mod}}, \text{PP, NP})$
$(shares_1,\ \text{Word: } shares,\ \text{Mod Gpar Rule: S} \rightarrow \text{VP} \rightarrow \text{VBD}_{\text{head}}, \text{NP}_{\text{mod}}, \text{PP, NP})$

Pairwise features generated for $(shares_1, bought_3) =$

$(shares_1, bought_3,\ \text{Dependency: VBD, NP, VP, Right, +Adj, -CC})$
$(shares_1, bought_3,\ \text{Rule: VP} \rightarrow \text{VBD}_{\text{head}}, \text{NP}_{\text{mod}}, \text{PP, NP})$
$(shares_1, bought_3,\ \text{Gpar Rule: S} \rightarrow \text{VP} \rightarrow \text{VBD}_{\text{head}}, \text{NP}_{\text{mod}}, \text{PP, NP})$

Figure 2: The features used in our model. We show the single–variable features produced for hidden value *shares*$_1$ and the pairwise features produced for hidden values $(\textit{shares}_1, \textit{bought}_3)$, in the context of the given parse fragment. Highest NT = highest nonterminal, Up Path = sequence of ancestor nonterminals, Down Path = sequence of headed nonterminals, Head Rule = rules headed by the word, Mod Rule = rule in which word acts as modifier, Head/Mod Gpar Rule = Head/Mod Rule plus grandparent nonterminal.

domains (in each case we give the model type that results from the definition—clustering, refinement, or pre–built ontology—in parentheses):

Lexical (Refinement) Each word is split into three sub–values. See Figure 2 for an example of features generated for this choice of domain.

Part–of–Speech (Clustering) The part–of–speech tag of each word is split into five sub–values. In Figure 2, the word *shares* would be assigned the domain $\{\text{NNS}_1, \ldots, \text{NNS}_5\}$, and the word *bought* would have the domain $\{\text{VBD}_1, \ldots, \text{VBD}_5\}$.

Highest Nonterminal (Clustering) The highest nonterminal to which each word propagates as a headword is split into five sub–values. In Figure 2 the word *bought* yields domain $\{\text{S}_1, \ldots, \text{S}_5\}$, while *in* yields $\{\text{PP}_1, \ldots, \text{PP}_5\}$.

Supersense (Pre–Built Ontology) We borrow the idea of using WordNet lexicographer filenames as broad "supersenses" from Ciaramita and Johnson (2003). For each word, we split each of its supersenses into three sub–supersenses. If no supersenses are available, we fall back to splitting the part–of–speech into five sub–values. For example, *shares* has the supersenses noun.possession, noun.act and noun.artifact, which yield the domain $\{\text{noun.possession}_1, \text{noun.act}_1, \text{noun.artifact}_1, \ldots \text{noun.possession}_3, \text{noun.act}_3, \text{noun.artifact}_3\}$. On the other hand, *in* does not have any WordNet supersenses, so it is assigned the domain $\{\text{IN}_1, \ldots, \text{IN}_5\}$.

4.2 The Final Feature Sets

We created eight feature sets by combining the four hidden–value domains above with two alternative definitions of dependency structures: standard head–modifier dependencies and "sibling dependencies." When using sibling dependencies, connections are established between the headwords of adjacent siblings. For instance, the head–modifier dependencies produced by the tree fragment in Figure 2 are $(\textit{bought}, \textit{shares})$, $(\textit{bought}, \textit{in})$, and $(\textit{bought}, \textit{yesterday})$, while the corresponding sibling dependencies are $(\textit{bought}, \textit{shares})$, $(\textit{shares}, \textit{in})$, and $(\textit{in}, \textit{yesterday})$.

4.3 Mixed Models

The different hidden–variable models display varying strengths and weaknesses. We created mixtures of different models using a weighted average:

$$\log p(t_{i,j}|s_i) = \sum_{m=1}^{M} \lambda_m \log p_m(t_{i,j}|s_i, \Theta_m) - Z(s_i)$$

where $Z(s_i)$ is a normalization constant that can be ignored, as it does not affect the ranking of parses. The λ_m weights are determined through optimization of parsing scores on a development set.

5 Experimental Results

We trained and tested the model on data from the Penn Treebank (Marcus et al., 1994). Candidate parses were produced by an N–best version of the Collins (1999) parser. Our training data consists of Treebank Sections 2–21, divided into a training corpus of 35,540 sentences and a development corpus of 3,676 sentences. In later experiments, we made use of a secondary development corpus of 1,914 sentences from Section 0. Sections 22–24, containing 5,455 sentences, were held out as the test set.

For each of the eight feature sets described in Section 4.2, we used the stochastic gradient descent

	Section 22		Section 23		Section 24		Total	
	LR	**LP**	**LR**	**LP**	**LR**	**LP**	**LR**	**LP**
C99	89.12	89.20	88.14	88.56	87.17	87.97	88.19	88.60
MIX	90.43	90.61	89.25	89.69	88.46	89.29	89.41	89.87
C2K	90.27	90.62	89.43	89.97	88.56	89.58	89.46	90.07
MIX+	90.57	90.79	89.80	90.27	88.78	89.73	89.78	90.29

Table 1: The results on Sections 22–24. **LR** = Labeled Recall, **LP** = Labeled Precision.

method to optimize the parameters of the model. We created various mixtures of the eight models using the weighted–average technique described in Section 4.3, testing the accuracy of each mixture on the secondary development set. Our final model was a mixture of three of the eight possible models: supersense hidden values with sibling trees, lexical hidden values with sibling trees, and highest nonterminal hidden values with normal head–modifier trees.

Our final tests evaluated four models. The two baseline models are the Collins (1999) base parser, C99, and the Collins (2000) reranker, C2K. The first new model is the MIX model, which is a combination of the C99 base model with the three models described above. The second new model, MIX+, is created by augmenting MIX with features from the method in C2K. Table 1 shows the results. The MIX model obtains an F–measure improvement of $\approx 1.25\%$ over the C99 baseline, an improvement that is comparable to the C2K reranker. The MIX+ model yields an improvement of $\approx 0.25\%$ beyond C2K.

We tested the significance of 8 comparisons corresponding to the results in Table 1 using the sign test[4]: we tested MIX vs. C99 on Sections 22, 23, and 24 individually, as well as on Sections 22–24 taken as a whole; we also tested MIX+ vs. C2K on these 4 test sets. Of the 8 comparisons, all showed significant improvements at the level $p \leq 0.01$ with the exception of one test, MIX+ vs. C2K on Section 24.

6 Discussion

6.1 Applying the Model to Other NLP Tasks

In this section, we discuss how hidden–variable models might be applied to other NLP problems, and in particular to structures other than parse trees. To summarize the model, the major components of the approach are as follows:

• We assume some set of candidate structures $t_{i,j}$, which are to be reranked by the model. Each structure $t_{i,j}$ has $n_{i,j}$ words $w_1, \ldots, w_{n_{i,j}}$, and each word w_k has a set $H_k(t_{i,j})$ of possible hidden values.

• We assume a graph $\mathbf{D}(t_{i,j})$ for each $t_{i,j}$ that defines possible interactions between hidden variables in the model. We assume some definition of local feature vectors, which consider either single hidden variables, or pairs of hidden variables that are connected by an edge in $\mathbf{D}(t_{i,j})$.

The approach can be instantiated in several ways when applying the model to other NLP tasks. We have already seen that by changing the definition of the hidden–value domains $H_k(t_{i,j})$, we can derive models with widely varying behavior. In addition, there is no requirement that the hidden variables only be associated with words in the structure; the hidden variables could be associated with other units. For example, in speech recognition hidden variables could be associated with phonemes rather than words, and in Chinese word segmentation, hidden variables could be associated with individual characters rather than words.

NLP tasks other than parsing involve structures $t_{i,j}$ that are not necessarily parse trees. For example, in speech recognition candidates are simply strings (utterances); in tagging tasks candidates are labeled sequences (e.g., sentences labeled with part–of–speech tag sequences); in machine translation candidate structures may be source–language/target–language pairs, along with alignment structures specifying the correspondence between words in the two languages. Sentences and labeled sequences are in a sense simplifications of the parsing case, where a natural choice for the underlying graph $\mathbf{D}(t_{i,j})$ would be an N^{th} order Markov structure, where each word depends on the previous N words. Machine translation alignments are a more interesting type of structure, where the choice of $\mathbf{D}(t_{i,j})$ might actually depend on the alignment between the two sentences.

As a final note, there is some flexibility in the choice of $\mathbf{D}(t_{i,j})$. In the case that $\mathbf{D}(t_{i,j})$ is a tree belief propagation is exact. In the more general case where $\mathbf{D}(t_{i,j})$ contains cycles, there are alternative algorithms that are either exact (Cowell et al., 1999) or approximate (Yedidia et al., 2003).

[4]The input to the sign test is a set of sentences with judgements for each sentence indicating whether model 1 gives a better parse than model 2, model 2 gives a better parse than model 1, or models 1 and 2 give equal quality parses. When using the sign test, for each sentence in question we calculate the F–measure at the sentence level for the two models being compared, deriving the required judgement from these scores.

6.2 Packed Representations and Locality

One natural extension of our reranker is to adapt it to candidate parses represented as a packed parse forest, so that it can operate on the base parser's full output instead of a limited N-best list. However, as we described in Section 3.1, our features are locally scoped with respect to hidden–variable interactions but unrestricted regarding information derived from the underlying candidate parses, which poses a problem for the use of packed representations. For instance, the Up/Down Path features (see Figure 2) enumerate the vertical sequences of nonterminals that extend above and below a given headword. We could restrict the features to local scope on the candidate parses, allowing dynamic–programming to be used to train the model with a packed representation. However, even with these restrictions, finding $\arg\max_t \sum_{\mathbf{h}} p(t, \mathbf{h} \mid s, \Theta)$ is NP–hard, and the Viterbi approximation $\arg\max_{t,\mathbf{h}} p(t, \mathbf{h} \mid s, \Theta)$ — or other approximations — would have to be used (see Matsuzaki et al. (2005)).

6.3 Empirical Analysis of the Hidden Values

Our model makes no assumptions about the interpretation of the hidden values assigned to words: during training, the model simply learns a distribution over global hidden–value assignments that is useful in improving the log–likelihood of the training data. Intuitively, however, we expect that the model will learn to make hidden–value assignments that are reasonable from a linguistic standpoint. In this section we describe some empirical observations concerning hidden values assigned by the model.

We established a corpus of parse trees with hidden–value annotations, as follows. First, we find the optimal parameters Θ^* on the training set. For every sentence s_i in the training set, we then use Θ^* to find t_i^*, the most probable candidate parse under the model. Finally, we use Θ^* to decode $\mathbf{h}_i^*$, the most probable global assignment of hidden values, for each parse tree t_i^*. We created a corpus of $(t_i^*, \mathbf{h}_i^*)$ pairs for the feature set defined by part–of–speech hidden–value domains and standard dependency structures. The remainder of this section describes trends for several of the most common part–of–speech categories in the corpus.

As a first example, consider the hidden values for the part–of–speech VB (infinitival verb). In the majority of cases, words tagged VB either modify a modal verb tagged MD (e.g., in *the new rate will/*MD *be/*VB *payable*) or the infinitival marker *to* (e.g., in *in an effort to streamline/*VB *bureaucracy*). The statistics of our corpus reflect this distinction. In 11,546 cases of the VB_1 hidden value, 10,652 cases modified *to*, and 81 cases modified modals tagged MD. In contrast, in 11,042 cases of the VB_2 value, the numbers were 8,354 and 599 for modification of modals and *to* respectively, showing the opposite preference. This polarization of hidden values allows modifiers to the VB (e.g., *payable* in *the new rate will be payable*) to be sensitive to whether the verb is modifying a modal or *to*.

In a related case, the hidden values for the part–of–speech TO (corresponding to the word *to*) also show that the model is learning useful structure. Consider cases where *to* heads a clause which may or may not have a subject (e.g., in *it expects* ⟨*its sales to remain steady*⟩ vs. *a proposal* ⟨*to ease reporting requirements*⟩). We find that for hidden values TO_1 and TO_5 together, 946 out of 976 cases have a subject. In contrast, for the hidden value TO_4, only 29 out of 10,148 cases have a subject. This splitting of the TO part–of–speech allows modifiers to *to*, or words modified by *to*, to be sensitive to the presence or absence of a subject in the clause headed by *to*.

Finally, consider the hidden values for the part–of–speech NNS (plural noun). In this case, the model distinguishes contexts where a plural noun acting as the head of a noun–phrase is or isn't modified by a post–modifier (such as a prepositional phrase or relative clause). For hidden value NNS_3, 12,003 out of the 12,664 instances in our corpus have a post–modifier, but for hidden value NNS_5, only 4,099 of the 39,763 occurrences have a post–modifier. Similar contextual effects were observed for other noun categories such as singular or proper nouns.

7 Conclusions and Future Research

The hidden–variable model is a novel method for representing NLP structures in the reranking framework. We can obtain versatile behavior from the model simply by manipulating the definition of the hidden–value domains, and we have experimented with models that emulate word clustering, word refinement, and mappings from words into an existing ontology. In the case of the parse reranking task, the hidden–variable model achieves reranking performance comparable to the reranking approach described by Collins (2000), and the two rerankers can be combined to yield an additive improvement.

Future work may consider the use of hidden–value domains with mixed contents, such as a domain that contains 3 refinement–oriented lexical values and 3 clustering–oriented part–of–speech values. These mixed values would allow the hidden–variable model to exploit interactions between clustering and refinement at the level of words and dependencies. Another area for future research is to investigate the use of unlabeled data within the approach, for example by making use of clusters derived from large amounts of unlabeled data (e.g., see Miller et al. (2004)). Finally, future work may apply the models to NLP tasks other than parsing.

Acknowledgements

We would like to thank Regina Barzilay, Igor Malioutov, and Luke Zettlemoyer for their many comments on the paper. We gratefully acknowledge the support of the National Science Foundation (under grants 0347631 and 0434222) and the DARPA/SRI CALO project (through subcontract No. 03-000215).

References

Daniel Bikel. 2000. A statistical model for parsing and word–sense disambiguation. In *Proceedings of EMNLP*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class–based n–gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Eugene Charniak and Mark Johnson. 2005. Coarse–to–fine n–best parsing and maxent discriminative reranking. In *Proceedings of the 43rd ACL*.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *EMNLP 2003*.

Stephen Clark and James R. Curran. 2004. Parsing the wsj using ccg and log–linear models. In *ACL*, pages 103–110.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL 2002*.

Michael Collins. 1999. *Head–Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th ICML*.

Michael Collins. 2002. Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *ACL 2002*.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification–based" grammars. In *Proceedings of the 37th ACL*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient–based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November.

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.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *ACL*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1993. Five papers on wordnet. Technical report, Stanford University.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT–NAACL*, pages 337–342.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th ACL*, pages 295–302.

Fernando C. N. Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st ACL*, pages 183–190.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2005. Conditional random fields for object recognition. In *NIPS 17*. MIT Press.

Adwait Ratnaparkhi, Salim Roukos, and R. Todd Ward. 1994. A maximum entropy model for parsing. In *ICSLP 1994*.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard S. Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the wall street journal using a lexical–functional grammar and discriminative estimation techniques. In *ACL*.

Libin Shen and Aravind K. Joshi. 2003. An svm–based voting algorithm with application to parse reranking. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the 7th CoNLL*, pages 9–16. Edmonton, Canada.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *HLT–NAACL*, pages 177–184.

Marilyn A. Walker, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *NAACL*.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss, 2003. *Exploring Artificial Intelligence in the New Millennium*, chapter 8: "Understanding Belief Propagation and Its Generalizations", pages 239–236. Science & Technology Books.

Disambiguation of Morphological Structure using a PCFG

Helmut Schmid
Institute for Natural Language Processing (IMS)
University of Stuttgart
Germany
schmid@ims.uni-stuttgart.de

Abstract

German has a productive morphology and allows the creation of complex words which are often highly ambiguous. This paper reports on the development of a head-lexicalized PCFG for the disambiguation of German morphological analyses. The grammar is trained on unlabeled data using the Inside-Outside algorithm. The parser achieves a precision of more than 68% on difficult test data, which is 23% more than the baseline obtained by randomly choosing one of the simplest analyses. Remarkable is the fact that precision drops to 52% without lexicalization.

1 Introduction

German words may be as complex as the following title of a bill: *Rindfleischetikettierungs-überwachungsaufgabenübertragungsgesetz* (law for the transfer of the task of controlling the labeling of beef). The complexity is due to the productive morphological processes of derivation (e.g. *Etikettierung* (labeling) = *Etikett* (label) + *ier* (derivational suffix) + *ung* (nominalization suffix)) and compounding (e.g. *Rindfleisch* (beef) = *Rind* (cattle) + *Fleisch* (meat)). For many words, there is more than one possible analysis. The German SMOR morphology (Schmid et al., 2004) e.g. generates 24 analyses for the word *Abteilungen*. If differences in the case feature are ignored, there are still six analyses, all of them plural:

- *Abt* (abbot) *Ei* (egg) *Lunge* (lung) *n* (plural inflectional ending)
- *Abt* (abbot) *ei* (abbot → abbey) *Lunge* (lung) *n* (plural inflectional ending)
- *Abt* (abbot) *eil* (hurry) *ung* (nominalization suffix) *en* (plural inflectional ending)
- *Abtei* (abbey) *Lunge* (lung) *n* (plural inflectional ending)
- *Abteilung* (department) *en* (plural inflectional ending)
- *ab* (separable verb prefix) *teil* (divide) *ung* (nominalization suffix) *en* (plural inflectional ending)

Here – and in many other cases, as well – the least complex analysis (defined as the number of derivation and compounding steps), namely the plural of *Abteilung* (department), is the correct one. This heuristic is not always successful, however. The word *Reisende* e.g. is analyzed as the compound of *Reis* (rice) + *Ende* (end), and alternately as the nominalization of the present participle *reisend* (traveling). The latter one is correct although it requires two derivational steps (formation of the participle plus nominalization), while the former requires only one compounding step.

The least complex analysis is not necessarily unique. One reason is, that German word forms are often ambiguous wrt. number, gender and case. The adjective *kleine* (small) e.g. receives 7 analyses by SMOR which differ only in the agreement features.

Another reason is that word forms are ambiguous wrt. the part of speech. The word *gerecht* e.g. is either an adjective (fair) or the past participle of the verb *rechen* (to rake). Similarly, the word *gerade* is either an adjective (straight) or an adverb (just). These ambiguities can be resolved based on the context e.g. with a part-of-speech tagger.

Other types of ambiguities are not disambiguated by the syntactic context because the morphosyntactic features are invariant. Compounds with three elements like *Sonderpreisliste*, for instance, are systematically ambiguous between a left-branching (list of special prices) and a right-branching structure (special price-list), but unambiguous regarding their part of speech and agreement features. Some word forms like *Schmerzfreiheit* (absence of pain) can either be analyzed as derivations (schmerzfrei-heit – "painless-ness") or as compounds (Schmerz-freiheit – "pain freedom"). Again, there is no difference in the morphosyntactic features. A further source of ambiguity are the stems: Consider the word *Mittelzuweisung* (allocation of resources). The compounding stem *mittel* could either originate from the adjective *mittel* (average) or from the noun *Mittel* (means). All these ambiguities are not resolvable by the syntactic context because their syntactic properties are identical. However, most of these words have a preferred reading. *Nah verkehrs zug* (commuter train) e.g. is likely to have a left-branching structure, whereas the correct analysis of *Computer bild schirm* (computer monitor) is right-branching.

Considering these features of German morphology, the following disambiguation strategy for morphological ambiguities is proposed: Frequent words should be manually disambiguated and the correct analysis/analyses should be explicitly stored in the lexicon. Ambiguities involving different morphosyntactic features should be resolved by a tagger or parser. The elimination of the remaining ambiguities, namely ambiguities of rare words which are not reflected by the morphosyntactic features, requires a different method. A general strategy is to generate the set of possible analyses, to rank them according to some criterion and to return the best analysis (or analyses). One very simple ranking criterion is the complexity of the analysis e.g. measured by the number of derivational and compounding steps. We will use this criterion as a baseline to which we compare our method.

Given an FST-based morphological analyzer and a training corpus consisting of manually disambiguated analyses, it is also possible to estimate transition probabilities for the finite state transducer and to disambiguate by choosing the most probable path through the transducer network for a given word. A drawback of this approach is the limitation of the type of analyses that finite state transducers can generate. A finite state transducer maps a regular language, the set of word forms, to another regular language, the set of analyses. Therefore it is not able to produce structured analyses as shown in figure 1 (for arbitrary depths). It also fails to represent non-local dependencies, like the one between *Vertrag* (contract) and *Lösung* (solution) in the second analysis of figure 1.

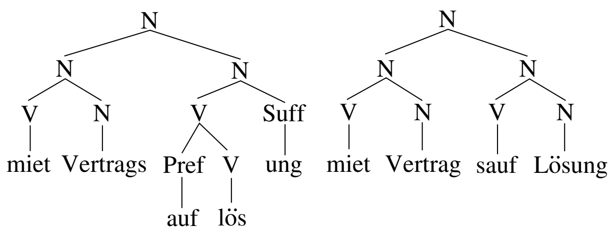


Figure 1: Two morphological analyses of the German word *Mietvertragsauflösung* (leasing contract cancellation); the first one is correct.

Given the limitations of weighted finite-state transducers, we propose to use a more powerful formalism, namely head-lexicalized probabilistic context-free grammars (Carroll and Rooth, 1998; Charniak, 1997) to rank the analyses. Context-free grammars have, of course, no difficulties to generate the analyses shown in figure 1. By assigning probabilities to the grammar rules, we obtain a probabilistic context-free grammar (PCFG) which allows the parser to distinguish between frequent and rare morphological constructions. Nouns e.g. are much more likely to be compounds than verbs. In head-lexicalized PCFGs (HL-PCFGs), the probability of a rule also depends on the lexical head of the constituent. HL-PCFGs are therefore able to learn that nouns headed by *Problem* (problem) are more likely to be compounds (e.g. *Schulprobleme* (school problems)) than nouns headed by *Samstag* (Saturday). Moreover, HL-PCFGs represent lexical dependencies like that between *Vertrag* and *Lösung* in figure 1.

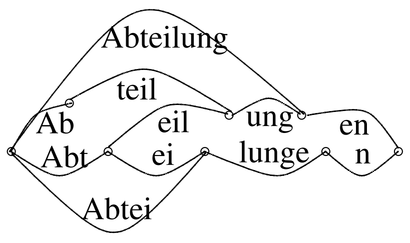


Figure 2: Morpheme lattice

In this paper, we present a HL-PCFG-based disambiguator for German. Using the SMOR morphological analyzer, the input words are first split into morpheme sequences and then analyzed with a HL-PCFG parser. Due to ambiguities, the parser's input is actually a lattice rather than a sequence (see the example in figure 2).

The rest of the paper is organized as follows: In section 2, we briefly review head-lexicalized PCFGs. Section 3 summarizes some important features of SMOR. The development of the grammar will be described in section 4. Section 5 explains the training strategy, and section 6 reports on the annotation of the test data. Section 7 presents the results of an evaluation, section 8 comments on related work, and section 9 summarizes the main points of the paper. Finally, section 10 gives an outlook on future work.

2 Head-Lexicalized PCFGs

A *head-lexicalized parse tree* is a parse tree in which each constituent is labeled with its category and its *lexical head*. The lexical head of a terminal symbol is the symbol itself and the lexical head of a non-terminal symbol is the lexical head of its (unique) head child.

In a head-lexicalized PCFG (HL-PCFG) (Carroll and Rooth, 1998; Charniak, 1997), one symbol on the right-hand side of each rule is marked as the head. A HL-PCFG assumes that (i) the probability of a rule depends on the category **and the lexical head** of the expanded constituent and (ii) that the lexical head of a non-head node depends on its own category, and the category and the lexical head of the parent node. The probability of a head-lexicalized parse tree is therefore:

$$p_{start}(cat(root))\ p_{start}(head(root)|cat(root)) *$$
$$\prod_{n \in N} p_{rule}(rule(n)|cat(n), head(n)) *$$
$$\prod_{n \in A} p_{head}(head(n)|cat(n), pcat(n), phead(n))$$

where

$root$ is the root node of the parse tree
$cat(n)$ is the syntactic category of node n
$head(n)$ is the lexical head of node n
$rule(n)$ is the grammar rule which expands node n
$pcat(n)$ is the syntactic category of the parent of n
$phead(n)$ is the lexical head of the parent of n

HL-PCFGs have a large number of parameters which need to estimated from training data. In order to avoid sparse data problems, the parameters usually have to be smoothed. HL-PCFGs can either be trained on labeled data (supervised training) or on unlabeled data (unsupervised training) using the Inside-Outside algorithm, an instance of the EM algorithm. Training on labeled data usually gives better results, but it requires a treebank which is expensive to create. In our experiments, we used unsupervised training with the LoPar parser which is available at http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/LoPar-en.html.

3 SMOR

SMOR (Schmid et al., 2004) is a German FST-based morphological analyzer which covers inflection, compounding, and prefix as well as suffix derivation. It builds on earlier work reported in (Schiller, 1996) and (Schmid et al., 2001).

SMOR uses features to represent derivation constraints. German derivational suffixes select their base in terms of part of speech, the stem type (derivation or compounding stem)[1], the origin (native, classical, foreign), and the structure (simplex, compound, prefix derivation, suffix derivation) of the stem which they combine with. This information is encoded with features. The German derivation suffix *lich* e.g. combines with a simplex derivation stem of a native noun to form an adjective. The feature constraints of *lich* are therefore (1) part of speech = NN (2) stem type = deriv (3) origin = native and (4) structure = simplex.

[1]Suffixes which combine with compounding stems historically evolved from compounding constructions.

4 The Grammar

The grammar used by the morphological disambiguator has a small set of rather general categories for prefixes (P), suffixes (S), uninflected base stems (B), uninflected base suffixes (SB), inflectional endings (F) and other morphemes (W). There is only one rule for compounding and prefix and suffix derivation, respectively, and two rules for the stem and suffix inflection. Additional rules introduce the start symbol TOP and generate special word forms like hyphenated (*Thomas-Mann-Straße*) or truncated words (*Vor-*). Overall, the base grammar has 13 rules. Inflection is always attached low in order to avoid spurious ambiguities. The part of speech is encoded as a feature.

Like SMOR, the grammar encodes derivation constraints with features. Number, gender and case are not encoded. Ambiguities in the agreement features are therefore not reflected in the parses which the grammar generates. This allows us to abstract away from this type of ambiguity which cannot be resolved without contextual information. If some application requires agreement information, it has to be reinserted after disambiguation.

The feature grammar is compiled into a context-free grammar with 1973 rules. In order to reduce the grammar size, the features for origin and complexity were not compiled out. Figure 3 shows a compounding rule (building a noun base stem from a noun compounding stem and a noun base stem), a suffix derivation rule (building an adjective base stem from a noun derivation stem and a derivation suffix), a prefix derivation rule (prefixing a verbal compounding stem) and two inflection rules (for the inflection of a noun and a nominal derivation suffix, respectively) from the resulting grammar. The quote symbol marks the head of a rule.

W.NN.base → W.NN.compound W.NN.base'
W.ADJ.base → W.NN.deriv S.NN.deriv.ADJ.base
W.V.compound → P.V W.V.compound'
W.NN.base → B.NN.base' F.NN
S.ADJ.deriv.NN.base → SB.ADJ.deriv.NN.base' F.NN

Figure 3: Some rules from the context-free grammar

The parser retrieves the categories of the morphemes from a lexicon which also contains information about the standard form of a morpheme. The representation of the morphemes returned by the FST-based word splitter is close to the surface form. Only capitalization is taken over from the standard form. The adjective *ursächlich* (causal), for instance, is split into *Ursäch* and *lich*. The lexicon assigns to *Ursäch* the category *W.NN.deriv* and the standard form *Ursache* (cause).

5 PCFG Training

PCFG training normally requires manually annotated training data. Because a treebank of German morphological analyses was not available, we decided to try unsupervised training using the Inside-Outside algorithm (Lari and Young, 1990). We worked with unlexicalized as well as head-lexicalized PCFGs (Carroll and Rooth, 1998; Charniak, 1997). The lexicalized models used the standard form of the morphemes (see the previous section) as heads.

The word list from a German 300 million word newspaper corpus was used as training data. From the 3.2 million tokens in the word list, SMOR successfully analyzed 2.3 million tokens which were used in the experiment. Training was either type-based (with each word form having the same weight) or token-based (with weights proportional to the frequency). We experimented with uniform and non-uniform initial distributions. In the uniform model, each rule had an initial frequency of 1 from which the probabilities were estimated. In the non-uniform model, the frequency of two classes of rules was increased to 1000. The first class are the rules which expand the start symbol TOP to an adjective or adverb, leading to a preference of these word classes over other word classes, in particular verbs. The second class is formed by rules generating inflectional endings, which induces a preference for simpler analyses.

6 Test Data

The test data was extracted from a corpus of the German newspaper *Die Zeit* which was not part of the training corpus. We prepared two different test corpora. The first test corpus (data1) consisted of 425 words extracted from a randomly selected part of the corpus. We only extracted words with at least one letter which were ambiguous (ignoring ambiguities in number, gender and case) and either nouns, verbs or adjectives and not from the beginning of a sentence. Duplicates were retained. The words were parsed and manually disambiguated. We looked at the context of a word, where this was necessary for disambiguation. Words without a correct analysis were deleted.

In order to obtain more information on the types of ambiguity and their frequency, 200 words were manually classified wrt. the class of the ambiguity. The following results were obtained:

- 39 words (25%) were ambiguous between an adjective and a verb like *gerecht* - "just" (adjective) vs. past participle of *rechen* (to rake).
- 28 words (18%) were ambiguous between a noun and a proper name like *Mann* - "man" vs. *Thomas Mann*
- 19 words were ambiguous between an adjective and an adverb like *gerade* - "straight" vs. "just" (adverb)
- 14 words (9%) showed a complex ambiguity involving derivation and compounding like the word *überlieferung* (tradition) which is either a nominalization of the prefix verb *überliefern* (to bequeath) or a compound of the stems *über* (over) and *Lieferung* (delivery).
- 13 words (8%) were compounds which were ambiguous between a left-branching and a right-branching structure like *Welt rekord höhe* (world record height)
- In 10 words (5%), there was an ambiguity between an adjective and a proper name or noun stem - as in *Höchstleistung* (maximum performance) where *höchst* can be derived from the proper name *Höchst* (a German city) or the superlative *höchst* (highest)
- 6 words (3%) showed a systematic ambiguity between an adjective and a noun caused by adding the suffix *er* to a city name, like *Moskauer* - "Moskau related" vs. "person from Moskau"
- Another 6 words were ambiguous between two different noun stems like *Halle* which is either singular form of *Halle* (hall) or the plural form of *Hall* (reverberation)

Overall 50% of the ambiguities involved a part-of-speech ambiguity.

The second set of test data (data2) was designed to contain only infrequent words which were not ambiguous wrt. part of speech. It was extracted from the same newspaper corpus. Here, we excluded words which were (1) sentence-initial (in order to avoid problems with capitalized words) (2) not analyzed by SMOR (3) ambiguous wrt. part of speech (4) from closed word classes or (5) simplex words. Furthermore, we extracted only words with more than one simplest[2] analysis, in order to make the test data more challenging. The extracted words were sorted by frequency and a block of 1000 word forms was randomly selected from the lower frequency range. All of them had occurred 4 times. We focussed on rare words because frequent words are better disambiguated manually and stored in a table (see the discussion in the introduction).

The 1000 selected word forms were parsed and manually disambiguated. 193 problematic words were deleted from the evaluation set because either (1) no analysis was correct (e.g. *Elsevier*, which was not analyzed as a proper name) or (2) there was a true ambiguity (e.g. *Rottweiler* which is either a dog breed or a person from the city of Rottweil or (3) the lemma was not unique (*Drehtür* (revolving door) could be lemmatized to *Drehtür* or *Drehtüre* with no difference in meaning.) or (4) several analyses were equivalent. The disambiguation was often difficult. Even among the words retained in the test set, there were many that we were not fully sure about. An example is the compound *Natur eis bahn* ("natural ice rink") which we decided to analyze as *Natur-Eisbahn* (nature ice-rink) rather than *Natureis-Bahn* (nature-ice rink).

7 Results

The parser was trained using the Inside-Outside algorithm. By default, (a) the initialization of the rule

[2]The complexity of an analysis is measured by the number of derivation and compounding steps.

probabilities was non-uniform as described in section 5, (b) training was based on tokens (i.e. the frequency of the training items was taken into account), and (c) all training iterations were lexicalized. Training was quite fast. One training iteration on 2.3 million word forms took about 10 minutes on a Pentium IV running at 3 GHz.

Figure 4 shows the exact match accuracy of the Viterbi parses depending on the number of training iterations, which ranges from 0 (the initial, untrained model) to 15. For comparison, a baseline result is shown which was obtained by selecting the set of simplest analyses and choosing one of them at random[3]. The baseline accuracy was 45.3%. The parsing accuracy of the default model jumps from a starting value of 41.8% for the untrained model (which is below the baseline) to 58.5% after a single training iteration. The peak performance is reached after 8 iterations with 65.4%. The average accuracy of the models obtained after 6-25 iterations is 65.1%.

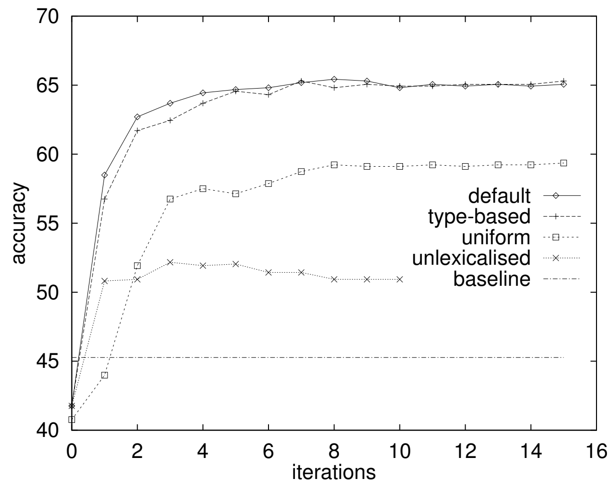


Figure 4: Exact match accuracy on data2

Results obtained with type-based training, where each word receives the same weight ignoring its frequency, were virtually identical to those of the default model. If the parser training was started with a uniform initial model, however, the accuracy dropped by about 6 percentage points. Figure 4 also shows that the performance of an unlexicalized PCFG is about 13% lower.

We also experimented with a combination of unlexicalized and lexicalized training. Lexicalized models have a huge number of parameters. Therefore, there is a large number of locally optimal parameter settings to which the unsupervised training can be attracted. Purely lexicalized training is likely to get stuck in a local optimum which is close to the starting point. Unlexicalized models, on the other hand, have fewer parameters, a smaller number of local optima and a smoother search space. Unlexicalized training is therefore more likely to reach a globally (near-)optimal point and provides a better starting point for the lexicalized training.

Figure 5 shows that initial unlexicalized training indeed improves the accuracy of the parser. With one iteration of unlexicalized training (see "unlex 1" in figure 5), the accuracy increased by about 3%. The maximum of 68.4% was reached after 4 iterations of lexicalized training. The results obtained with 2 iterations of unlexicalized training were very similar. With 3 iterations, the performance dropped almost to the level of the default model. It seems that some of the general preferences learned during unlexicalized training are so strong after three iterations that they cannot be overruled anymore by the lexeme-specific preferences learned in the lexicalized training.

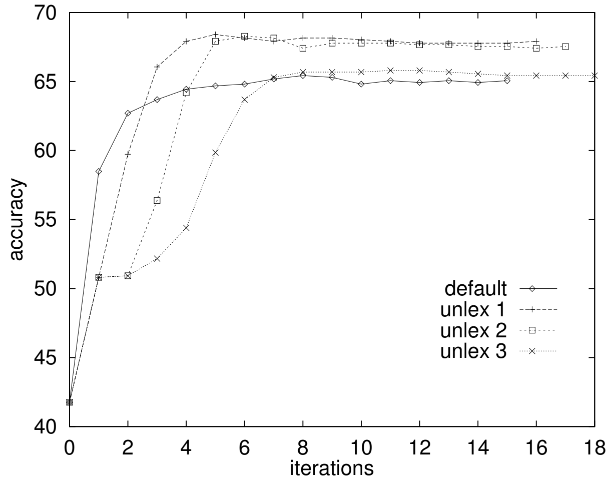


Figure 5: Results on data2 with 0 (default), 1, 2, or 3 iterations of unlexicalized training, followed by lexicalized training

In order to assess the parsing results qualitatively,

[3]In fact, we counted a word with n simplest analyses as $1/n$ correct instead of actually selecting one analysis at random, in order to avoid a dependency of the baseline result on the random number generation.

100 parsing errors of version “unlex 2” were randomly selected and inspected. It turned out that the parser always preferred right-branching structures over left-branching structures in complex compounds with three or more elements, which resulted in 57 errors caused by left-branching structures. Grammars trained without the initial unlexicalized training showed no systematic preference for right-branching structures. In the test data, left-branching structures were two times more frequent than right-branching structures.

29 disambiguation errors resulted from selecting the wrong stem although the structure of the analysis was otherwise correct. In the word *Rechtskonstruktion* (legal construction), for instance, the first element *Rechts* was derived from the adjective *rechts* (right) rather than the noun *Recht* (law). Similarly, the adjective *quelloffen* (open-source) was derived from the verb *quellen* (to swell) rather than the noun *Quelle* (source).

Six errors involved a combination of compounding and suffix derivation (e.g. the word *Flugbegleiterin* (stewardess)). The parser preferred the analysis where the derivation is applied first (*Flug-Begleiterin* (flight attendant[female])), whereas in the gold standard analysis, the compound is formed first (*Flugbegleiter-in* (steward-ess).

In order to better understand the benefits of unlexicalized training, we also examined the differences between the best model obtained with one iteration of unlexicalized (unlex1), and the best model obtained without unlexicalized training (default).

30 cases involved left-branching vs. right-branching compounds. The unlex1 model showed a higher preference for right-branching structures than the default model, but produced also left-branching structures (unlike the model unlex2). In 15 of the 30 cases, unlex1 correctly decided for a right-branching structure; in 13 cases, unlex1 was wrong with proposing a right-branching structure. In two cases, unlex1 correctly predicted a left-branching structure and the default model predicted a right-branching structure.

32 differences were caused by lexical ambiguities. In 24 cases, only one stem was ambiguous. 15 times unlex1 was right (e.g. *Moskaureise* - Moskow trip[sg] vs. Moskow rice[pl]) and nine times the default model was right (e.g. *Jodtabletten* - iodine pill vs. iodine tablet). In 8 cases, two morphemes were involved in the ambiguity. In all these cases, unlex1 generated the correct analysis (e.g. *Sportraum* - “sport room” vs. “Spor[name] dream”).

Nine ambiguities involved the length of verb prefixes. Six times, unlex1 correctly decided for a longer prefix (e.g. *gegenüber-stehen* (to face) instead of *gegen-überstehen* (to “counter-survive”).

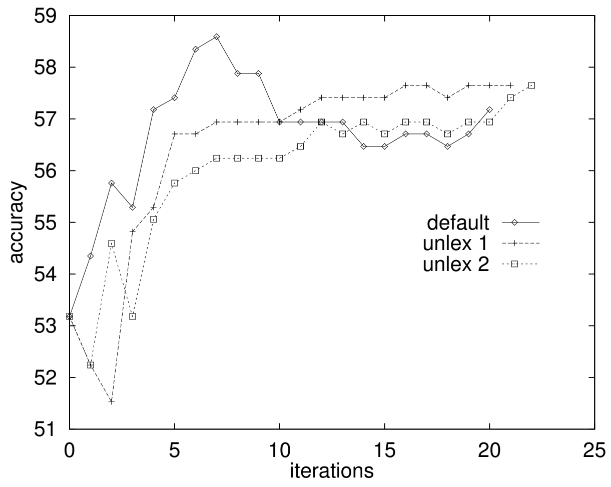


Figure 6: Accuracy on data1 after 0, 1, or 2 iterations of unlexicalized training followed by lexicalized training

In another experiment, we tested the parser on the first test data set (data1) where simplex words, part-of-speech ambiguities, frequent words and repeated occurrences were not removed. The baseline accuracy on this data was 43.75%. Figure 6 shows the results obtained with different numbers of unlexicalized training iterations analogous to figure 5. Strictly lexicalized training produced the best results, here. The maximal accuracy was 58.59% which was obtained after 7 iterations. In contrast to the experiments on data2, the accuracy decreased by more than 1.5% when the training was continued. As said in the introduction, we think that part-of-speech ambiguities are better resolved by a part-of-speech tagger and that frequent words can be disambiguated manually.

8 Related Work

New methods are often first developed for English and later adapted to other languages. This might explain why morphological disambiguation has been so rarely addressed in the past: English morphology is seldom ambiguous except for noun compounds.

We are not aware of any work on the disambiguation of morphological analyses which is directly comparable to ours. Mark Lauer (1995) only considered English noun compounds and applied a different disambiguation strategy based on word association scores.

Koehn and Knight (2003) proposed a splitting method for German compounds and showed that it improves statistical machine translation. Compounds are split into smaller pieces (which have to be words themselves) if the geometric mean of the word frequencies of the pieces is higher than the frequency of the compound. Information from a bilingual corpus is used to improve the splitting accuracy.

Andreas Eisele (unpublished work) implemented a statistical disambiguator for German based on weighted finite-state transducers as described in the introduction. However, his system fails to represent and disambiguate the ambiguities observed in compounds with three or more elements and similar constructions with structural ambiguities.

9 Summary

We presented a disambiguation method for German morphological analyses which is based on a head-lexicalized probabilistic context-free grammar. The words are split into morpheme lattices by a finite state morphology, and then parsed with the probabilistic context-free grammar. The grammar was trained on unlabeled data using the Inside-Outside algorithm and evaluated on 807 manually disambiguated analyses of infrequent words. Different training strategies have been compared. A combination of one iteration of unlexicalized training and four iterations of lexicalized training returned the best results with over 68% exact match accuracy, compared to a baseline of 45% which was obtained by randomly choosing one of the minimally complex analyses. Without lexicalization, the accuracy dropped by 15 percentage points, indicating that lexicalization is essential for morphological disambiguation.

10 Future Work

There are several starting points for improvement. Guidelines should be developed for the manual annotation of data in order to make it less dependent on the annotator's intuitions. More data should be annotated to create a treebank of morphological analyses. Given such a treebank, the parser could be trained on labeled data or on a combination of labeled and unlabeled data, which presumably would further increase the parsing accuracy.

References

Glenn Carroll and Mats Rooth. 1998. Valence induction with a head-lexicalized PCFG. In *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing*, Granada, Spain.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word Statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, Menlo Parc.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

K. Lari and S. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computation Speech and Language Processing*, 4:35–56.

Mark Lauer. 1995. Corpus statistics meet the noun compound: Some empirical results. In *Proceedings of the 33rd Annual Meeting of the ACL, Massachusetts Institute of Technology*, pages 47–54, Cambridge, Mass. electronically available at http://xxx.lanl.gov/abs/cmp-lg/9504033.

Anne Schiller. 1996. Deutsche Flexions- und Kompositionsmorphologie mit PC-KIMMO. In Roland Hausser, editor, *Proceedings, 1. Morpholympics, Erlangen, 7./8. Mrz 1994*, Tübingen. Niemeyer.

Tanja Schmid, Anke Ldeling, Bettina Suberlich, Ulrich Heid, and Bernd Mbius. 2001. DeKo: Ein System zur Analyse komplexer Wrter. In *GLDV - Jahrestagung 2001*, pages 49–57.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, volume 4, pages 1263–1266, Lisbon, Portugal.

Non-projective Dependency Parsing using Spanning Tree Algorithms

Ryan McDonald **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
{ryantm,pereira}@cis.upenn.edu

Kiril Ribarov **Jan Hajič**
Institute of Formal and Applied Linguistics
Charles University
{ribarov,hajic}@ufal.ms.mff.cuni.cz

Abstract

We formalize weighted dependency parsing as searching for maximum spanning trees (MSTs) in directed graphs. Using this representation, the parsing algorithm of Eisner (1996) is sufficient for searching over all projective trees in $O(n^3)$ time. More surprisingly, the representation is extended naturally to non-projective parsing using Chu-Liu-Edmonds (Chu and Liu, 1965; Edmonds, 1967) MST algorithm, yielding an $O(n^2)$ parsing algorithm. We evaluate these methods on the Prague Dependency Treebank using online large-margin learning techniques (Crammer et al., 2003; McDonald et al., 2005) and show that MST parsing increases efficiency and accuracy for languages with non-projective dependencies.

1 Introduction

Dependency parsing has seen a surge of interest lately for applications such as relation extraction (Culotta and Sorensen, 2004), machine translation (Ding and Palmer, 2005), synonym generation (Shinyama et al., 2002), and lexical resource augmentation (Snow et al., 2004). The primary reasons for using dependency structures instead of more informative lexicalized phrase structures is that they are more efficient to learn and parse while still encoding much of the predicate-argument information needed in applications.

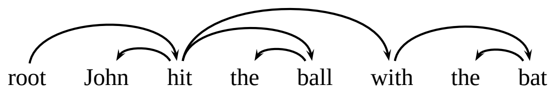


Figure 1: An example dependency tree.

Dependency representations, which link words to their arguments, have a long history (Hudson, 1984). Figure 1 shows a dependency tree for the sentence *John hit the ball with the bat*. We restrict ourselves to *dependency tree* analyses, in which each word depends on exactly one parent, either another word or a dummy root symbol as shown in the figure. The tree in Figure 1 is *projective*, meaning that if we put the words in their linear order, preceded by the root, the edges can be drawn above the words without crossings, or, equivalently, a word and its descendants form a contiguous substring of the sentence.

In English, projective trees are sufficient to analyze most sentence types. In fact, the largest source of English dependency trees is automatically generated from the Penn Treebank (Marcus et al., 1993) and is by convention exclusively projective. However, there are certain examples in which a non-projective tree is preferable. Consider the sentence *John saw a dog yesterday which was a Yorkshire Terrier*. Here the relative clause *which was a Yorkshire Terrier* and the object it modifies (the *dog*) are separated by an adverb. There is no way to draw the dependency tree for this sentence in the plane with no crossing edges, as illustrated in Figure 2. In languages with more flexible word order than English, such as German, Dutch and Czech, non-projective dependencies are more frequent. Rich inflection systems reduce reliance on word order to express

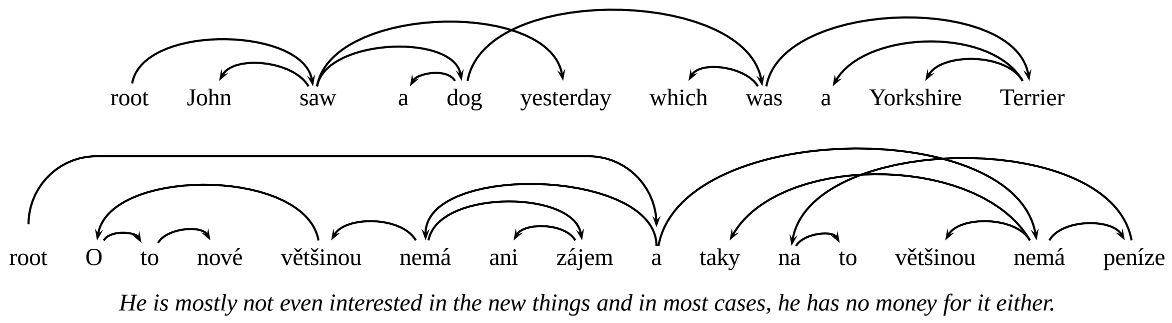

root John saw a dog yesterday which was a Yorkshire Terrier
root O to nové většinou nemá ani zájem a taky na to většinou nemá peníze
He is mostly not even interested in the new things and in most cases, he has no money for it either.

Figure 2: Non-projective dependency trees in English and Czech.

grammatical relations, allowing non-projective dependencies that we need to represent and parse efficiently. A non-projective example from the Czech Prague Dependency Treebank (Hajič et al., 2001) is also shown in Figure 2.

Most previous dependency parsing models have focused on projective trees, including the work of Eisner (1996), Collins et al. (1999), Yamada and Matsumoto (2003), Nivre and Scholz (2004), and McDonald et al. (2005). These systems have shown that accurate projective dependency parsers can be automatically learned from parsed data. However, non-projective analyses have recently attracted some interest, not only for languages with freer word order but also for English. In particular, Wang and Harper (2004) describe a broad coverage non-projective parser for English based on a hand-constructed constraint dependency grammar rich in lexical and syntactic information. Nivre and Nilsson (2005) presented a parsing model that allows for the introduction of non-projective edges into dependency trees through learned edge transformations within their memory-based parser. They test this system on Czech and show improved accuracy relative to a projective parser. Our approach differs from those earlier efforts in searching optimally and efficiently the full space of non-projective trees.

The main idea of our method is that dependency parsing can be formalized as the search for a maximum spanning tree in a directed graph. This formalization generalizes standard projective parsing models based on the Eisner algorithm (Eisner, 1996) to yield efficient $O(n^2)$ exact parsing methods for non-projective languages like Czech. Using this spanning tree representation, we extend the work of McDonald et al. (2005) on online large-margin discriminative training methods to non-projective dependencies.

The present work is related to that of Hirakawa (2001) who, like us, reduces the problem of dependency parsing to spanning tree search. However, his parsing method uses a branch and bound algorithm that is exponential in the worst case, even though it appears to perform reasonably in limited experiments. Furthermore, his work does not adequately address learning or measure parsing accuracy on held-out data.

Section 2 describes an edge-based factorization of dependency trees and uses it to equate dependency parsing to the problem of finding maximum spanning trees in directed graphs. Section 3 outlines the online large-margin learning framework used to train our dependency parsers. Finally, in Section 4 we present parsing results for Czech. The trees in Figure 1 and Figure 2 are untyped, that is, edges are not partitioned into types representing additional syntactic information such as grammatical function. We study untyped dependency trees mainly, but edge types can be added with simple extensions to the methods discussed here.

2 Dependency Parsing and Spanning Trees

2.1 Edge Based Factorization

In what follows, $\boldsymbol{x} = x_1 \cdots x_n$ represents a generic input sentence, and $\boldsymbol{y}$ represents a generic dependency tree for sentence $\boldsymbol{x}$. Seeing $\boldsymbol{y}$ as the set of tree edges, we write $(i, j) \in \boldsymbol{y}$ if there is a dependency in $\boldsymbol{y}$ from word x_i to word x_j.

In this paper we follow a common method of factoring the score of a dependency tree as the sum of the scores of all edges in the tree. In particular, we define the score of an edge to be the dot product between a high dimensional feature representation of the edge and a weight vector,

$$s(i, j) = \mathbf{w} \cdot \mathbf{f}(i, j)$$

Thus the score of a dependency tree $\boldsymbol{y}$ for sentence $\boldsymbol{x}$ is,

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} s(i, j) = \sum_{(i,j)\in\boldsymbol{y}} \mathbf{w} \cdot \mathbf{f}(i, j)$$

Assuming an appropriate feature representation as well as a weight vector **w**, dependency parsing is the task of finding the dependency tree $\boldsymbol{y}$ with highest score for a given sentence $\boldsymbol{x}$.

For the rest of this section we assume that the weight vector **w** is known and thus we know the score $s(i, j)$ of each possible edge. In Section 3 we present a method for learning the weight vector.

2.2 Maximum Spanning Trees

We represent the generic directed graph $G = (V, E)$ by its vertex set $V = \{v_1, \dots, v_n\}$ and set $E \subseteq [1 : n] \times [1 : n]$ of pairs (i, j) of directed edges $v_i \to v_j$. Each such edge has a score $s(i, j)$. Since G is directed, $s(i, j)$ does not necessarily equal $s(j, i)$. A *maximum spanning tree* (MST) of G is a tree $\boldsymbol{y} \subseteq E$ that maximizes the value $\sum_{(i,j)\in\boldsymbol{y}} s(i, j)$ such that every vertex in V appears in $\boldsymbol{y}$. The maximum *projective* spanning tree of G is constructed similarly except that it can only contain projective edges relative to some total order on the vertices of G. The MST problem for directed graphs is also known as the maximum arborescence problem.

For each sentence $\boldsymbol{x}$ we define the directed graph $G_{\boldsymbol{x}} = (V_{\boldsymbol{x}}, E_{\boldsymbol{x}})$ given by

$$\begin{aligned} V_{\boldsymbol{x}} &= \{x_0 = \text{root}, x_1, \dots, x_n\} \\ E_{\boldsymbol{x}} &= \{(i, j) : i \neq j, (i, j) \in [0 : n] \times [1 : n]\} \end{aligned}$$

That is, $G_{\boldsymbol{x}}$ is a graph with the sentence words and the dummy root symbol as vertices and a directed edge between every pair of distinct words and from the root symbol to every word. It is clear that dependency trees for $\boldsymbol{x}$ and spanning trees for $G_{\boldsymbol{x}}$ coincide, since both kinds of trees are required to be rooted at the dummy root and reach all the words in the sentence. Hence, finding a (projective) dependency tree with highest score is equivalent to finding a maximum (projective) spanning tree in $G_{\boldsymbol{x}}$.

```
Chu-Liu-Edmonds($G, s$)
    Graph $G = (V, E)$
    Edge weight function $s : E \to \mathbb{R}$
1.  Let $M = \{(x^*, x) : x \in V, x^* = \arg\max_{x'} s(x', x)\}$
2.  Let $G_M = (V, M)$
3.  If $G_M$ has no cycles, then it is an MST: return $G_M$
4.  Otherwise, find a cycle $C$ in $G_M$
5.  Let $G_C$ = contract$(G, C, s)$
6.  Let $\boldsymbol{y}$ = Chu-Liu-Edmonds$(G_C, s)$
7.  Find a vertex $x \in C$ s. t. $(x', x) \in \boldsymbol{y}, (x'', x) \in C$
8.  return $\boldsymbol{y} \cup C - \{(x'', x)\}$
contract($G = (V, E), C, s$)
1.  Let $G_C$ be the subgraph of $G$ excluding nodes in $C$
2.  Add a node $c$ to $G_C$ representing cycle $C$
3.  For $x \in V - C : \exists_{x' \in C}(x', x) \in E$
      Add edge $(c, x)$ to $G_C$ with
        $s(c, x) = \max_{x' \in C} s(x', x)$
4.  For $x \in V - C : \exists_{x' \in C}(x, x') \in E$
      Add edge $(x, c)$ to $G_C$ with
        $s(x, c) = \max_{x' \in C} [s(x, x') - s(a(x'), x') + s(C)]$
          where $a(v)$ is the predecessor of $v$ in $C$
          and $s(C) = \sum_{v \in C} s(a(v), v)$
5.  return $G_C$
```

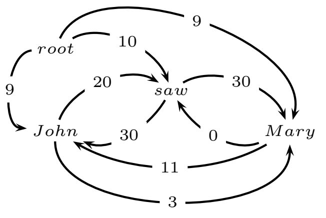
Figure 3: Chu-Liu-Edmonds algorithm for finding maximum spanning trees in directed graphs.

2.2.1 Non-projective Trees

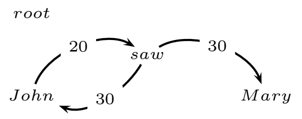
To find the highest scoring non-projective tree we simply search the entire space of spanning trees with no restrictions. Well-known algorithms exist for the less general case of finding spanning trees in undirected graphs (Cormen et al., 1990).

Efficient algorithms for the directed case are less well known, but they exist. We will use here the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), sketched in Figure 3 following Leonidas (2003). Informally, the algorithm has each vertex in the graph greedily select the incoming edge with highest weight. If a tree results, it must be the maximum spanning tree. If not, there must be a cycle. The procedure identifies a cycle and contracts it into a single vertex and recalculates edge weights going into and out of the cycle. It can be shown that a maximum spanning tree on the contracted graph is equivalent to a maximum spanning tree in the original graph (Leonidas, 2003). Hence the algorithm can recursively call itself on the new graph. Naively, this algorithm runs in $O(n^3)$ time since each recursive call takes $O(n^2)$ to find the highest incoming edge for each word and to contract the graph. There are at most $O(n)$ recursive calls since we cannot contract the graph more then n times. However, Tarjan (1977) gives an efficient implementation of the algorithm with $O(n^2)$ time complexity for dense graphs, which is what we need here.

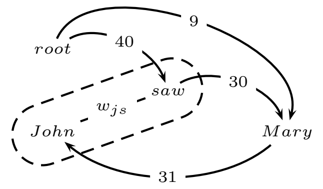
To find the highest scoring non-projective tree for a sentence, $\boldsymbol{x}$, we simply construct the graph $G_{\boldsymbol{x}}$ and run it through the Chu-Liu-Edmonds algorithm. The resulting spanning tree is the best non-projective dependency tree. We illustrate here the application of the Chu-Liu-Edmonds algorithm to dependency parsing on the simple example $\boldsymbol{x} =$ *John saw Mary*, with directed graph representation $G_{\boldsymbol{x}}$,



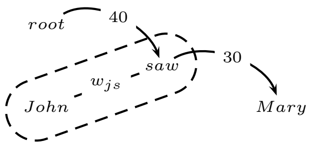
The first step of the algorithm is to find, for each word, the highest scoring incoming edge



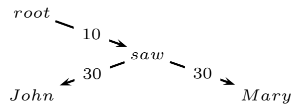
If the result were a tree, it would have to be the maximum spanning tree. However, in this case we have a cycle, so we will contract it into a single node and recalculate edge weights according to Figure 3.



The new vertex w_{js} represents the contraction of vertices *John* and *saw*. The edge from w_{js} to *Mary* is 30 since that is the highest scoring edge from any vertex in w_{js}. The edge from *root* into w_{js} is set to 40 since this represents the score of the best spanning tree originating from *root* and including only the vertices in w_{js}. The same leads to the edge from *Mary* to w_{js}. The fundamental property of the Chu-Liu-Edmonds algorithm is that an MST in this graph can be transformed into an MST in the original graph (Leonidas, 2003). Thus, we recursively call the algorithm on this graph. Note that we need to keep track of the real endpoints of the edges into and out of w_{js} for reconstruction later. Running the algorithm, we must find the best incoming edge to all words



This is a tree and thus the MST of this graph. We now need to go up a level and reconstruct the graph. The edge from w_{js} to *Mary* originally was from the word *saw*, so we include that edge. Furthermore, the edge from *root* to w_{js} represented a tree from *root* to *saw* to *John*, so we include all those edges to get the final (and correct) MST,



A possible concern with searching the entire space of spanning trees is that we have not used any syntactic constraints to guide the search. Many languages that allow non-projectivity are still primarily projective. By searching all possible non-projective trees, we run the risk of finding extremely bad trees. We address this concern in Section 4.

2.2.2 Projective Trees

It is well known that projective dependency parsing using edge based factorization can be handled with the Eisner algorithm (Eisner, 1996). This algorithm has a runtime of $O(n^3)$ and has been employed successfully in both generative and discriminative parsing models (Eisner, 1996; McDonald et al., 2005). Furthermore, it is trivial to show that the Eisner algorithm solves the maximum projective spanning tree problem.

The Eisner algorithm differs significantly from the Chu-Liu-Edmonds algorithm. First of all, it is a bottom-up dynamic programming algorithm as opposed to a greedy recursive one. A bottom-up algorithm is necessary for the projective case since it must maintain the nested structural constraint, which is unnecessary for the non-projective case.

2.3 Dependency Trees as MSTs: Summary

In the preceding discussion, we have shown that natural language dependency parsing can be reduced to finding maximum spanning trees in directed graphs. This reduction results from edge-based factorization and can be applied to projective languages with the Eisner parsing algorithm and non-projective languages with the Chu-Liu-Edmonds maximum spanning tree algorithm. The only remaining problem is how to learn the weight vector **w**.

A major advantage of our approach over other dependency parsing models is its uniformity and simplicity. By viewing dependency structures as spanning trees, we have provided a general framework for parsing trees for both projective and non-projective languages. Furthermore, the resulting parsing algorithms are more efficient than lexicalized phrase structure approaches to dependency parsing, allowing us to search the entire space without any pruning. In particular the non-projective parsing algorithm based on the Chu-Liu-Edmonds MST algorithm provides *true* non-projective parsing. This is in contrast to other non-projective methods, such as that of Nivre and Nilsson (2005), who implement non-projectivity in a *pseudo-projective* parser with edge transformations. This formulation also dispels the notion that non-projective parsing is "harder" than projective parsing. In fact, it is easier since non-projective parsing does not need to enforce the non-crossing constraint of projective trees. As a result, non-projective parsing complexity is just $O(n^2)$, against the $O(n^3)$ complexity of the Eisner dynamic programming algorithm, which by construction enforces the non-crossing constraint.

3 Online Large Margin Learning

In this section, we review the work of McDonald et al. (2005) for online large-margin dependency parsing. As usual for supervised learning, we assume a training set $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^T$, consisting of pairs of a sentence $\boldsymbol{x}_t$ and its correct dependency tree $\boldsymbol{y}_t$. In what follows, $\mathrm{dt}(\boldsymbol{x})$ denotes the set of possible dependency trees for sentence $\boldsymbol{x}$.

The basic idea is to extend the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2003) to learning with structured outputs, in the present case dependency trees. Figure 4 gives pseudo-code for the MIRA algorithm as presented by McDonald et al. (2005). An online learning algorithm considers a single training instance at each update to **w**. The auxiliary vector **v** accumulates the successive values of **w**, so that the final weight vector is the *average* of the weight vectors after each iteration. This averaging effect has been shown to help overfitting (Collins, 2002).

```
Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^T$
1. $\mathbf{w}_0 = 0;\ \mathbf{v} = 0;\ i = 0$
2. for $n : 1..N$
3.   for $t : 1..T$
4.     $\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\|$
       s.t. $s(\boldsymbol{x}_t, \boldsymbol{y}_t) - s(\boldsymbol{x}_t, \boldsymbol{y}') \geq L(\boldsymbol{y}_t, \boldsymbol{y}'), \forall \boldsymbol{y}' \in \mathrm{dt}(\boldsymbol{x}_t)$
5.     $\mathbf{v} = \mathbf{v} + \mathbf{w}^{(i+1)}$
6.     $i = i + 1$
7. $\mathbf{w} = \mathbf{v}/(N * T)$
```

Figure 4: MIRA learning algorithm.

On each update, MIRA attempts to keep the new weight vector as close as possible to the old weight vector, subject to correctly classifying the instance under consideration with a margin given by the loss of the incorrect classifications. For dependency trees, the loss of a tree is defined to be the number of words with incorrect parents relative to the correct tree. This is closely related to the Hamming loss that is often used for sequences (Taskar et al., 2003).

For arbitrary inputs, there are typically exponentially many possible parses and thus exponentially many margin constraints in line 4 of Figure 4.

3.1 Single-best MIRA

One solution for the exponential blow-up in number of trees is to relax the optimization by using only the single margin constraint for the tree with the highest score, $s(\boldsymbol{x}, \boldsymbol{y})$. The resulting online update (to be inserted in Figure 4, line 4) would then be:

$$\begin{aligned}
&\min\ \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\
&\text{s.t.}\quad s(\boldsymbol{x}_t, \boldsymbol{y}_t) - s(\boldsymbol{x}_t, \boldsymbol{y}') \geq L(\boldsymbol{y}_t, \boldsymbol{y}') \\
&\text{where } \boldsymbol{y}' = \arg\max_{y'} s(\boldsymbol{x}_t, \boldsymbol{y}')
\end{aligned}$$

McDonald et al. (2005) used a similar update with k constraints for the k highest-scoring trees, and showed that small values of k are sufficient to achieve the best accuracy for these methods. However, here we stay with a single best tree because k-best extensions to the Chu-Liu-Edmonds algorithm are too inefficient (Hou, 1996).

This model is related to the averaged perceptron algorithm of Collins (2002). In that algorithm, the single highest scoring tree (or structure) is used to update the weight vector. However, MIRA aggressively updates **w** to maximize the margin between the correct tree and the highest scoring tree, which has been shown to lead to increased accuracy.

3.2 Factored MIRA

It is also possible to exploit the structure of the output space and factor the exponential number of margin constraints into a polynomial number of local constraints (Taskar et al., 2003; Taskar et al., 2004). For the directed maximum spanning tree problem, we can factor the output by edges to obtain the following constraints:

$$\begin{aligned}&\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\ &\text{s.t. } s(l,j) - s(k,j) \geq 1 \\ &\forall (l,j) \in \boldsymbol{y}_t, (k,j) \notin \boldsymbol{y}_t\end{aligned}$$

This states that the weight of the correct incoming edge to the word x_j and the weight of all other incoming edges must be separated by a margin of 1. It is easy to show that when all these constraints are satisfied, the correct spanning tree and all incorrect spanning trees are separated by a score at least as large as the number of incorrect incoming edges. This is because the scores for all the correct arcs cancel out, leaving only the scores for the errors causing the difference in overall score. Since each single error results in a score increase of at least 1, the entire score difference must be at least the number of errors. For sequences, this form of factorization has been called local lattice preference (Crammer et al., 2004). Let n be the number of nodes in graph $G_{\boldsymbol{x}}$. Then the number of constraints is $O(n^2)$, since for each node we must maintain $n-1$ constraints.

The factored constraints are in general more restrictive than the original constraints, so they may rule out the optimal solution to the original problem. McDonald et al. (2005) examines briefly factored MIRA for projective English dependency parsing, but for that application, k-best MIRA performs as well or better, and is much faster to train.

4 Experiments

We performed experiments on the Czech Prague Dependency Treebank (PDT) (Hajič, 1998; Hajič et al., 2001). We used the predefined training, development and testing split of this data set. Furthermore, we used the automatically generated POS tags that are provided with the data. Czech POS tags are very complex, consisting of a series of slots that may or may not be filled with some value. These slots represent lexical and grammatical properties such as standard POS, case, gender, and tense. The result is that Czech POS tags are rich in information, but quite sparse when viewed as a whole. To reduce sparseness, our features rely only on the reduced POS tag set from Collins et al. (1999). The number of features extracted from the PDT training set was $13,450,672$, using the feature set outlined by McDonald et al. (2005).

Czech has more flexible word order than English and as a result the PDT contains non-projective dependencies. On average, 23% of the sentences in the training, development and test sets have at least one non-projective dependency. However, less than 2% of total edges are actually non-projective. Therefore, handling non-projective edges correctly has a relatively small effect on overall accuracy. To show the effect more clearly, we created two Czech data sets. The first, Czech-A, consists of the entire PDT. The second, Czech-B, includes only the 23% of sentences with at least one non-projective dependency. This second set will allow us to analyze the effectiveness of the algorithms on non-projective material. We compared the following systems:

1. **COLL1999:** The projective lexicalized phrase-structure parser of Collins et al. (1999).
2. **N&N2005:** The pseudo-projective parser of Nivre and Nilsson (2005).
3. **McD2005:** The projective parser of McDonald et al. (2005) that uses the Eisner algorithm for both training and testing. This system uses k-best MIRA with k=5.
4. **Single-best MIRA:** In this system we use the Chu-Liu-Edmonds algorithm to find the best dependency tree for Single-best MIRA training and testing.
5. **Factored MIRA:** Uses the quadratic set of constraints based on edge factorization as described in Section 3.2. We use the Chu-Liu-Edmonds algorithm to find the best tree for the test data.

4.1 Results

Results are shown in Table 1. There are two main metrics. The first and most widely recognized is *Accuracy*, which measures the number of words that correctly identified their parent in the tree. *Complete* measures the number of sentences in which the resulting tree was completely correct.

Clearly, there is an advantage in using the Chu-Liu-Edmonds algorithm for Czech dependency pars-

	Czech-A		Czech-B	
	Accuracy	**Complete**	**Accuracy**	**Complete**
COLL1999	82.8	-	-	-
N&N2005	80.0	31.8	-	-
McD2005	83.3	31.3	74.8	0.0
Single-best MIRA	84.1	32.2	81.0	**14.9**
Factored MIRA	**84.4**	**32.3**	**81.5**	14.3

Table 1: Dependency parsing results for Czech. Czech-B is the subset of Czech-A containing only sentences with at least one non-projective dependency.

ing. Even though less than 2% of all dependencies are non-projective, we still see an absolute improvement of up to 1.1% in overall accuracy over the projective model. Furthermore, when we focus on the subset of data that only contains sentences with at least one non-projective dependency, the effect is amplified. Another major improvement here is that the Chu-Liu-Edmonds non-projective MST algorithm has a parsing complexity of $O(n^2)$, versus the $O(n^3)$ complexity of the projective Eisner algorithm, which in practice leads to improvements in parsing time. The results also show that in terms of *Accuracy*, factored MIRA performs better than single-best MIRA. However, for the factored model, we do have $O(n^2)$ margin constraints, which results in a significant increase in training time over single-best MIRA. Furthermore, we can also see that the MST parsers perform favorably compared to the more powerful lexicalized phrase-structure parsers, such as those presented by Collins et al. (1999) and Zeman (2004) that use expensive $O(n^5)$ parsing algorithms. We should note that the results in Collins et al. (1999) are different then reported here due to different training and testing data sets.

One concern raised in Section 2.2.1 is that searching the entire space of non-projective trees could cause problems for languages that are primarily projective. However, as we can see, this is not a problem. This is because the model sets its weights with respect to the parsing algorithm and will disfavor features over unlikely non-projective edges.

Since the space of projective trees is a subset of the space of non-projective trees, it is natural to wonder how the Chu-Liu-Edmonds parsing algorithm performs on projective data since it is asymptotically better than the Eisner algorithm. Table 2 shows the results for English projective dependency trees extracted from the Penn Treebank (Marcus et al., 1993) using the rules of Yamada and Matsumoto (2003).

	English	
	Accuracy	**Complete**
McD2005	**90.9**	**37.5**
Single-best MIRA	90.2	33.2
Factored MIRA	90.2	32.3

Table 2: Dependency parsing results for English using spanning tree algorithms.

This shows that for projective data sets, training and testing with the Chu-Liu-Edmonds algorithm is worse than using the Eisner algorithm. This is not surprising since the Eisner algorithm uses the a priori knowledge that all trees are projective.

5 Discussion

We presented a general framework for parsing dependency trees based on an equivalence to maximum spanning trees in directed graphs. This framework provides natural and efficient mechanisms for parsing both projective and non-projective languages through the use of the Eisner and Chu-Liu-Edmonds algorithms. To learn these structures we used online large-margin learning (McDonald et al., 2005) that empirically provides state-of-the-art performance for Czech.

A major advantage of our models is the ability to naturally model non-projective parses. Non-projective parsing is commonly considered more difficult than projective parsing. However, under our framework, we show that the opposite is actually true that non-projective parsing has a lower asymptotic complexity. Using this framework, we presented results showing that the non-projective model outperforms the projective model on the Prague Dependency Treebank, which contains a small number of non-projective edges.

Our method requires a tree score that decomposes according to the edges of the dependency tree. One might hope that the method would generalize to include features of larger substructures. Unfortunately, that would make the search for the best tree intractable (Höffgen, 1993).

Acknowledgments

We thank Lillian Lee for bringing an important missed connection to our attention, and Koby Crammer for his help with learning algorithms. This work has been supported by NSF ITR grants 0205448 and 0428193.

References

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proc. ACL*.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

T.H. Cormen, C.E. Leiserson, and R.L. Rivest. 1990. *Introduction to Algorithms*. MIT Press/McGraw-Hill.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*.

K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2003. Online passive aggressive algorithms. In *Proc. NIPS*.

K. Crammer, R. McDonald, and F. Pereira. 2004. New large margin algorithms for structured prediction. In *Learning with Structured Outputs Workshop (NIPS)*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. ACL*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. ACL*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.

J. Hajič, E. Hajicova, P. Pajas, J. Panevova, P. Sgall, and B. Vidova Hladka. 2001. The Prague Dependency Treebank 1.0 CDROM. Linguistics Data Consortium Cat. No. LDC2001T10.

J. Hajič. 1998. Building a syntactically annotated corpus: The Prague dependency treebank. *Issues of Valency and Meaning*, pages 106–132.

H. Hirakawa. 2001. Semantic dependency analysis method for Japanese based on optimum tree search algorithm. In *Proc. of PACLING*.

Klaus-U. Höffgen. 1993. Learning and robust learning of product distributions. In *Proceedings of COLT'93*, pages 77–83.

W. Hou. 1996. Algorithm for fi nding the fi rst k shortest arborescences of a digraph. *Mathematica Applicata*, 9(1):1–4.

R. Hudson. 1984. *Word Grammar*. Blackwell.

G. Leonidas. 2003. Arborescence optimization problems solvable by Edmonds' algorithm. *Theoretical Computer Science*, 301:427 – 437.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. ACL*.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of english text. In *Proc. COLING*.

Y. Shinyama, S. Sekine, K. Sudo, and R. Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proc. HLT*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2004*.

R.E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7:25–35.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proc. NIPS*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proc. EMNLP*.

W. Wang and M. P. Harper. 2004. A statistical constraint dependency grammar (CDG) parser. In *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.

D. Zeman. 2004. *Parsing with a Statistical Dependency Model*. Ph.D. thesis, Univerzita Karlova, Praha.

Making Computers Laugh: Investigations in Automatic Humor Recognition

Rada Mihalcea
Department of Computer Science
University of North Texas
Denton, TX, 76203, USA
rada@cs.unt.edu

Carlo Strapparava
Istituto per la Ricerca Scientifica e Tecnologica
ITC – irst
I-38050, Povo, Trento, Italy
strappa@itc.it

Abstract

Humor is one of the most interesting and puzzling aspects of human behavior. Despite the attention it has received in fields such as philosophy, linguistics, and psychology, there have been only few attempts to create computational models for humor recognition or generation. In this paper, we bring empirical evidence that computational approaches can be successfully applied to the task of humor recognition. Through experiments performed on very large data sets, we show that automatic classification techniques can be effectively used to distinguish between humorous and non-humorous texts, with significant improvements observed over apriori known baselines.

1 Introduction

... pleasure has probably been the main goal all along. But I hesitate to admit it, because computer scientists want to maintain their image as hard-working individuals who deserve high salaries. Sooner or later society will realize that certain kinds of hard work are in fact admirable even though they are more fun than just about anything else. (Knuth, 1993)

Humor is an essential element in personal communication. While it is merely considered a way to induce amusement, humor also has a positive effect on the mental state of those using it and has the ability to improve their activity. Therefore computational humor deserves particular attention, as it has the potential of changing computers into a creative and motivational tool for human activity (Stock et al., 2002; Nijholt et al., 2003).

Previous work in computational humor has focused mainly on the task of humor generation (Stock and Strapparava, 2003; Binsted and Ritchie, 1997), and very few attempts have been made to develop systems for automatic humor recognition (Taylor and Mazlack, 2004). This is not surprising, since, from a computational perspective, humor recognition appears to be significantly more subtle and difficult than humor generation.

In this paper, we explore the applicability of computational approaches to the recognition of verbally expressed humor. In particular, we investigate whether automatic classification techniques are a viable approach to distinguish between humorous and non-humorous text, and we bring empirical evidence in support of this hypothesis through experiments performed on very large data sets.

Since a deep comprehension of humor in all of its aspects is probably too ambitious and beyond the existing computational capabilities, we chose to restrict our investigation to the type of humor found in *one-liners*. A one-liner is a short sentence with comic effects and an interesting linguistic structure: simple syntax, deliberate use of rhetoric devices (e.g. alliteration, rhyme), and frequent use of creative language constructions meant to attract the readers attention. While longer jokes can have a relatively complex narrative structure, a one-liner must produce the humorous effect "in one shot", with very few words. These characteristics make this type of humor particularly suitable for use in an automatic learning setting, as the humor-producing features are guaranteed to be present in the first (and only) sentence.

We attempt to formulate the humor-recognition problem as a traditional classification task, and feed positive (humorous) and negative (non-humorous) examples to an automatic classifier. The humorous data set consists of one-liners collected from the Web using an automatic bootstrapping process. The non-humorous data is selected such that it is structurally and stylistically similar to the one-liners. Specifically, we use three different negative data sets: (1) Reuters news titles; (2) proverbs; and (3) sentences from the British National Corpus (BNC). The classification results are encouraging, with accuracy figures ranging from 79.15% (One-liners/BNC) to 96.95% (One-liners/Reuters). Regardless of the non-humorous data set playing the role of negative examples, the performance of the automatically learned humor-recognizer is always significantly better than apriori known baselines.

The remainder of the paper is organized as follows. We first describe the humorous and non-humorous data sets, and provide details on the Web-based bootstrapping process used to build a very large collection of one-liners. We then show experimental results obtained on these data sets using several heuristics and two different text classifiers. Finally, we conclude with a discussion and directions for future work.

2 Humorous and Non-humorous Data Sets

To test our hypothesis that automatic classification techniques represent a viable approach to humor recognition, we needed in the first place a data set consisting of both humorous (positive) and non-humorous (negative) examples. Such data sets can be used to automatically *learn* computational models for humor recognition, and at the same time *evaluate* the performance of such models.

2.1 Humorous Data

For reasons outlined earlier, we restrict our attention to one-liners, short humorous sentences that have the characteristic of producing a comic effect in very few words (usually 15 or less). The one-liners humor style is illustrated in Table 1, which shows three examples of such one-sentence jokes.

It is well-known that large amounts of training data have the potential of improving the accuracy of the learning process, and at the same time provide insights into how increasingly larger data sets can affect the classification precision. The manual construction of a very large one-liner data set may be however problematic, since most Web sites or mailing lists that make available such jokes do not usually list more than 50–100 one-liners. To tackle this problem, we implemented a Web-based bootstrapping algorithm able to automatically collect a large number of one-liners starting with a short *seed* list, consisting of a few one-liners manually identified.

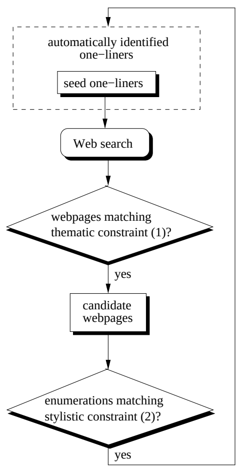

automatically identified one−liners
seed one−liners
Web search
webpages matching thematic constraint (1)?
yes
candidate webpages
enumerations matching stylistic constraint (2)?
yes

Figure 1: Web-based bootstrapping of one-liners.

The bootstrapping process is illustrated in Figure 1. Starting with the seed set, the algorithm automatically identifies a list of webpages that include at least one of the seed one-liners, via a simple search performed with a Web search engine. Next, the webpages found in this way are HTML parsed, and additional one-liners are automatically identified and added to the seed set. The process is repeated several times, until enough one-liners are collected.

An important aspect of any bootstrapping algorithm is the set of constraints used to steer the process and prevent as much as possible the addition of noisy entries. Our algorithm uses: (1) a *thematic* constraint applied to the theme of each webpage; and (2) a *structural* constraint, exploiting HTML annotations indicating text of similar genre.

The first constraint is implemented using a set of keywords of which at least one has to appear in the URL of a retrieved webpage, thus potentially limiting the content of the webpage to a theme related to that keyword. The set of keywords used in the current implementation consists of six words that explicitly indicate humor-related content: *oneliner*, *one-liner*, *humor*, *humour*, *joke*,

One-liners
Take my advice; I don't use it anyway.
I get enough exercise just pushing my luck.
Beauty is in the eye of the beer holder.
Reuters titles
Trocadero expects tripling of revenues.
Silver fi xes at two-month high, but gold lags.
Oil prices slip as refi ners shop for bargains.
BNC sentences
They were like spirits, and I loved them.
I wonder if there is some contradiction here.
The train arrives three minutes early.
Proverbs
Creativity is more important than knowledge.
Beauty is in the eye of the beholder.
I believe no tales from an enemy's tongue.

Table 1: Sample examples of one-liners, Reuters titles, BNC sentences, and proverbs.

funny. For example, *http://www.berro.com/Jokes* or *http://www.mutedfaith.com/funny/life.htm* are the URLs of two webpages that satisfy this constraint.

The second constraint is designed to exploit the HTML structure of webpages, in an attempt to identify enumerations of texts that include the seed one-liner. This is based on the hypothesis that enumerations typically include texts of similar genre, and thus a list including the seed one-liner is likely to include additional one-line jokes. For instance, if a seed one-liner is found in a webpage preceded by the HTML tag `<li>` (i.e. "list item"), other lines found in the same enumeration preceded by the same tag are also likely to be one-liners.

Two iterations of the bootstrapping process, started with a small seed set of ten one-liners, resulted in a large set of about 24,000 one-liners. After removing the duplicates using a measure of string similarity based on the longest common subsequence metric, we were left with a final set of approximately 16,000 one-liners, which are used in the humor-recognition experiments. Note that since the collection process is automatic, noisy entries are also possible. Manual verification of a randomly selected sample of 200 one-liners indicates an average of 9% potential noise in the data set, which is within reasonable limits, as it does not appear to significantly impact the quality of the learning.

2.2 Non-humorous Data

To construct the set of negative examples required by the humor-recognition models, we tried to identify collections of sentences that were non-humorous, but similar in structure and composition to the one-liners. We do not want the automatic classifiers to learn to distinguish between humorous and non-humorous examples based simply on text length or obvious vocabulary differences. Instead, we seek to enforce the classifiers to identify humor-specific features, by supplying them with negative examples similar in most of their aspects to the positive examples, but different in their comic effect.

We tested three different sets of negative examples, with three examples from each data set illustrated in Table 1. All non-humorous examples are enforced to follow the same length restriction as the one-liners, i.e. one sentence with an average length of 10–15 words.

1. *Reuters* titles, extracted from news articles published in the Reuters newswire over a period of one year (8/20/1996 – 8/19/1997) (Lewis et al., 2004). The titles consist of short sentences with simple syntax, and are often phrased to catch the readers attention (an effect similar to the one rendered by one-liners).
2. *Proverbs* extracted from an online proverb collection. Proverbs are sayings that transmit, usually in one short sentence, important facts or experiences that are considered true by many people. Their property of being condensed, but memorable sayings make them very similar to the one-liners. In fact, some one-liners attempt to reproduce proverbs, with a comic effect, as in e.g. *"Beauty is in the eye of the beer holder"*, derived from *"Beauty is in the eye of the beholder"*.
3. *British National Corpus (BNC)* sentences, extracted from BNC – a balanced corpus covering different styles, genres and domains. The sentences were selected such that they were similar in content with the one-liners: we used an information retrieval system implementing a vectorial model to identify the BNC sentence most similar to each of the 16,000 one-liners[1]. Unlike the Reuters titles or the proverbs, the BNC sentences have typically no added creativity. However, we decided to add this set of negative examples to our experimental setting, in order

[1]The sentence most similar to a one-liner is identifi ed by running the one-liner against an index built for all BNC sentences with a length of 10–15 words. We use a *tf.idf* weighting scheme and a cosine similarity measure, as implemented in the Smart system (*ftp.cs.cornell.edu/pub/smart*)

to observe the level of difficulty of a humor-recognition task when performed with respect to simple text.

To summarize, the humor recognition experiments rely on data sets consisting of humorous (positive) and non-humorous (negative) examples. The positive examples consist of 16,000 one-liners automatically collected using a Web-based bootstrapping process. The negative examples are drawn from: (1) Reuters titles; (2) Proverbs; and (3) BNC sentences.

3 Automatic Humor Recognition

We experiment with automatic classification techniques using: (a) heuristics based on humor-specific stylistic features (alliteration, antonymy, slang); (b) content-based features, within a learning framework formulated as a typical text classification task; and (c) combined stylistic and content-based features, integrated in a stacked machine learning framework.

3.1 Humor-Specific Stylistic Features

Linguistic theories of humor (Attardo, 1994) have suggested many *stylistic features* that characterize humorous texts. We tried to identify a set of features that were both significant and feasible to implement using existing machine readable resources. Specifically, we focus on alliteration, antonymy, and adult slang, which were previously suggested as potentially good indicators of humor (Ruch, 2002; Bucaria, 2004).

Alliteration. Some studies on humor appreciation (Ruch, 2002) show that structural and phonetic properties of jokes are at least as important as their content. In fact one-liners often rely on the reader's awareness of attention-catching sounds, through linguistic phenomena such as alliteration, word repetition and rhyme, which produce a comic effect even if the jokes are not necessarily meant to be read aloud. Note that similar rhetorical devices play an important role in wordplay jokes, and are often used in newspaper headlines and in advertisement. The following one-liners are examples of jokes that include one or more alliteration chains:

Veni, Vidi, Visa: I came, I saw, I did a little shopping.
Infants don't enjoy infancy like adults do adultery.

To extract this feature, we identify and count the number of alliteration/rhyme chains in each example in our data set. The chains are automatically extracted using an index created on top of the CMU pronunciation dictionary[2].

Antonymy. Humor often relies on some type of incongruity, opposition or other forms of apparent contradiction. While an accurate identification of all these properties is probably difficult to accomplish, it is relatively easy to identify the presence of *antonyms* in a sentence. For instance, the comic effect produced by the following one-liners is partly due to the presence of antonyms:

A clean desk is a sign of a cluttered desk drawer.
Always try to be modest and be proud of it!

The lexical resource we use to identify antonyms is WORDNET (Miller, 1995), and in particular the *antonymy* relation among nouns, verbs, adjectives and adverbs. For adjectives we also consider an indirect antonymy via the *similar-to* relation among adjective synsets. Despite the relatively large number of *antonymy* relations defined in WORDNET, its coverage is far from complete, and thus the *antonymy* feature cannot always be identified. A deeper semantic analysis of the text, such as word sense disambiguation or domain disambiguation, could probably help detecting other types of semantic opposition, and we plan to exploit these techniques in future work.

Adult slang. Humor based on adult slang is very popular. Therefore, a possible feature for humor-recognition is the detection of sexual-oriented lexicon in the sentence. The following represent examples of one-liners that include such slang:

The sex was so good that even the neighbors had a cigarette.
Artificial Insemination: procreation without recreation.

To form a lexicon required for the identification of this feature, we extract from WORDNET DOMAINS[3] all the synsets labeled with the domain SEXUALITY. The list is further processed by removing all words with high polysemy (≥ 4). Next, we check for the presence of the words in this lexicon in each sentence in the corpus, and annotate them accordingly. Note that, as in the case of antonymy, WORDNET coverage is not complete, and the *adult slang* feature cannot always be identified.

Finally, in some cases, all three features (alliteration,

[2] Available at *http://www.speech.cs.cmu.edu/cgi-bin/cmudict*

[3] WORDNET DOMAINS assigns each synset in WORDNET with one or more "domain" labels, such as SPORT, MEDICINE, ECONOMY. See *http://wndomains.itc.it*.

antonymy, adult slang) are present in the same sentence, as for instance the following one-liner:

Behind every great$_{al}$ *man*$_{ant}$ *is a great*$_{al}$ *woman*$_{ant}$*, and behind every great*$_{al}$ *woman*$_{ant}$ *is some guy staring at her behind*$_{sl}$*!*

3.2 Content-based Learning

In addition to stylistic features, we also experimented with *content-based features*, through experiments where the humor-recognition task is formulated as a traditional text classification problem. Specifically, we compare results obtained with two frequently used text classifiers, Naïve Bayes and Support Vector Machines, selected based on their performance in previously reported work, and for their diversity of learning methodologies.

Naïve Bayes. The main idea in a Naïve Bayes text classifier is to estimate the probability of a category given a document using joint probabilities of words and documents. Naïve Bayes classifiers assume word independence, but despite this simplification, they perform well on text classification. While there are several versions of Naïve Bayes classifiers (variations of multinomial and multivariate Bernoulli), we use the multinomial model, previously shown to be more effective (McCallum and Nigam, 1998).

Support Vector Machines. Support Vector Machines (SVM) are binary classifiers that seek to find the hyperplane that best separates a set of positive examples from a set of negative examples, with maximum margin. Applications of SVM classifiers to text categorization led to some of the best results reported in the literature (Joachims, 1998).

4 Experimental Results

Several experiments were conducted to gain insights into various aspects related to an automatic humor recognition task: classification accuracy using stylistic and content-based features, learning rates, impact of the type of negative data, impact of the classification methodology.

All evaluations are performed using stratified tenfold cross validations, for accurate estimates. The baseline for all the experiments is 50%, which represents the classification accuracy obtained if a label of "humorous" (or "non-humorous") would be assigned by default to all the examples in the data set. Experiments with uneven class distributions were also performed, and are reported in section 4.4.

4.1 Heuristics using Humor-specific Features

In a first set of experiments, we evaluated the classification accuracy using stylistic humor-specific features: alliteration, antonymy, and adult slang. These are numerical features that act as heuristics, and the only parameter required for their application is a threshold indicating the minimum value admitted for a statement to be classified as humorous (or non-humorous). These thresholds are learned automatically using a decision tree applied on a small subset of humorous/non-humorous examples (1000 examples). The evaluation is performed on the remaining 15,000 examples, with results shown in Table 2[4].

	One-liners	One-liners	One-liners
Heuristic	Reuters	BNC	Proverbs
Alliteration	74.31%	59.34%	53.30%
Antonymy	55.65%	51.40%	50.51%
Adult slang	52.74%	52.39%	50.74%
ALL	76.73%	60.63%	53.71%

Table 2: Humor-recognition accuracy using alliteration, antonymy, and adult slang.

Considering the fact that these features represent *stylistic* indicators, the style of Reuters titles turns out to be the most different with respect to one-liners, while the style of proverbs is the most similar. Note that for all data sets the alliteration feature appears to be the most useful indicator of humor, which is in agreement with previous linguistic findings (Ruch, 2002).

4.2 Text Classification with Content Features

The second set of experiments was concerned with the evaluation of content-based features for humor recognition. Table 3 shows results obtained using the three different sets of negative examples, with the Naïve Bayes and SVM text classifiers. Learning curves are plotted in Figure 2.

	One-liners	One-liners	One-liners
Classifier	Reuters	BNC	Proverbs
Naïve Bayes	96.67%	73.22%	84.81%
SVM	96.09%	77.51%	84.48%

Table 3: Humor-recognition accuracy using Naïve Bayes and SVM text classifiers.

[4]We also experimented with decision trees learned from a larger number of examples, but the results were similar, which confirms our hypothesis that these features are heuristics, rather than learnable properties that improve their accuracy with additional training data.

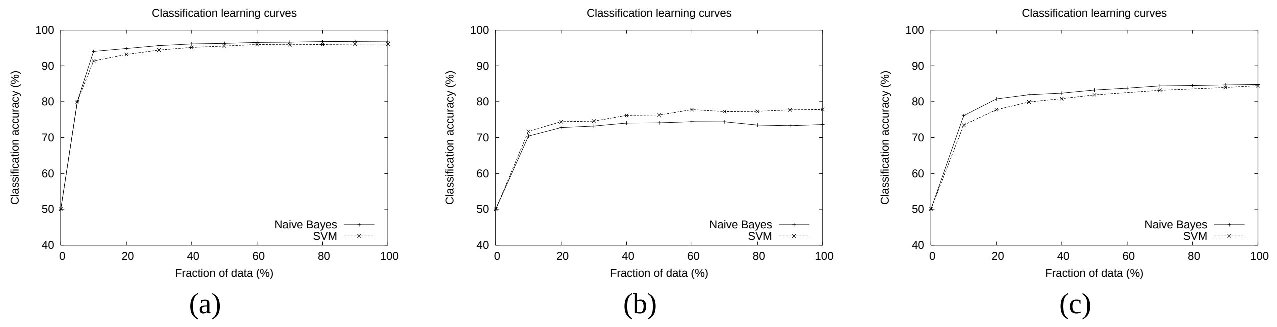


Figure 2: Learning curves for humor-recognition using text classification techniques, with respect to three different sets of negative examples: (a) Reuters; (b) BNC; (c) Proverbs.

Once again, the content of Reuters titles appears to be the most different with respect to one-liners, while the BNC sentences represent the most similar data set. This suggests that joke content tends to be very similar to regular text, although a reasonably accurate distinction can still be made using text classification techniques. Interestingly, proverbs can be distinguished from one-liners using content-based features, which indicates that despite their stylistic similarity (see Table 2), proverbs and one-liners deal with different topics.

4.3 Combining Stylistic and Content Features

Encouraged by the results obtained in the first two experiments, we designed a third experiment that attempts to jointly exploit stylistic and content features for humor recognition. The feature combination is performed using a stacked learner, which takes the output of the text classifier, joins it with the three humor-specific features (alliteration, antonymy, adult slang), and feeds the newly created feature vectors to a machine learning tool. Given the relatively large gap between the performance achieved with content-based features (text classification) and stylistic features (humor-specific heuristics), we decided to implement the second learning stage in the stacked learner using a memory based learning system, so that low-performance features are not eliminated in the favor of the more accurate ones[5]. We use the Timbl memory based learner (Daelemans et al., 2001), and evaluate the classification using a stratified ten-fold cross validation. Table 4 shows the results obtained in this experiment, for the three different data sets.

One-liners Reuters	One-liners BNC	One-liners Proverbs
96.95%	79.15%	84.82%

Table 4: Humor-recognition accuracy for combined learning based on stylistic and content features.

Combining classifiers results in a statistically significant improvement ($p < 0.0005$, paired t-test) with respect to the best individual classifier for the One-liners/Reuters and One-liners/BNC data sets, with relative error rate reductions of 8.9% and 7.3% respectively. No improvement is observed for the One-liners/Proverbs data set, which is not surprising since, as shown in Table 2, proverbs and one-liners cannot be clearly differentiated using stylistic features, and thus the addition of these features to content-based features is not likely to result in an improvement.

4.4 Discussion

The results obtained in the automatic classification experiments reveal the fact that computational approaches represent a viable solution for the task of humor-recognition, and good performance can be achieved using classification techniques based on stylistic and content features.

Despite our initial intuition that one-liners are most similar to other creative texts (e.g. Reuters titles, or the sometimes almost identical proverbs), and thus the learning task would be more difficult in relation to these data sets, comparative experimental results show that in fact it is more difficult to distinguish humor with respect to regular text (e.g. BNC

[5]Using a decision tree learner in a similar stacked learning experiment resulted into a flat tree that takes a classifi cation decision based exclusively on the content feature, ignoring completely the remaining stylistic features.

sentences). Note however that even in this case the combined classifier leads to a classification accuracy that improves significantly over the apriori known baseline.

An examination of the content-based features learned during the classification process reveals interesting aspects of the humorous texts. For instance, one-liners seem to constantly make reference to human-related scenarios, through the frequent use of words such as *man*, *woman*, *person*, *you*, *I*. Similarly, humorous texts seem to often include negative word forms, such as the negative verb forms *doesn't*, *isn't*, *don't*, or negative adjectives like *wrong* or *bad*. A more extensive analysis of content-based humor-specific features is likely to reveal additional humor-specific content features, which could also be used in studies of humor generation.

In addition to the three negative data sets, we also performed an experiment using a corpus of arbitrary sentences randomly drawn from the three negative sets. The humor recognition with respect to this negative mixed data set resulted in 63.76% accuracy for stylistic features, 77.82% for content-based features using Naïve Bayes and 79.23% using SVM. These figures are comparable to those reported in Tables 2 and 3 for One-liners/BNC, which suggests that the experimental results reported in the previous sections do not reflect a bias introduced by the negative data sets, since similar results are obtained when the humor recognition is performed with respect to arbitrary negative examples.

As indicated in section 2.2, the negative examples were selected structurally and stylistically similar to the one-liners, making the humor recognition task more difficult than in a real setting. Nonetheless, we also performed a set of experiments where we made the task even harder, using uneven class distributions. For each of the three types of negative examples, we constructed a data set using 75% non-humorous examples and 25% humorous examples. Although the baseline in this case is higher (75%), the automatic classification techniques for humor-recognition still improve over this baseline. The stylistic features lead to a classification accuracy of 87.49% (One-liners/Reuters), 77.62% (One-liners/BNC), and 76.20% (One-liners/Proverbs), and the content-based features used in a Naïve Bayes classifier result in accuracy figures of 96.19% (One-liners/Reuters), 81.56% (One-liners/BNC), and 87.86% (One-liners/Proverbs).

Finally, in addition to classification accuracy, we were also interested in the variation of classification performance with respect to data size, which is an aspect particularly relevant for directing future research. Depending on the shape of the learning curves, one could decide to concentrate future work either on the acquisition of larger data sets, or toward the identification of more sophisticated features. Figure 2 shows that regardless of the type of negative data, there is significant learning only until about 60% of the data (i.e. about 10,000 positive examples, and the same number of negative examples). The rather steep ascent of the curve, especially in the first part of the learning, suggests that humorous and non-humorous texts represent well distinguishable types of data. An interesting effect can be noticed toward the end of the learning, where for both classifiers the curve becomes completely flat (One-liners/Reuters, One-liners/Proverbs), or it even has a slight drop (One-liners/BNC). This is probably due to the presence of noise in the data set, which starts to become visible for very large data sets[6]. This plateau is also suggesting that more data is not likely to help improve the quality of an automatic humor-recognizer, and more sophisticated features are probably required.

5 Related Work

While humor is relatively well studied in scientific fields such as linguistics (Attardo, 1994) and psychology (Freud, 1905; Ruch, 2002), to date there is only a limited number of research contributions made toward the construction of computational humour prototypes.

One of the first attempts is perhaps the work described in (Binsted and Ritchie, 1997), where a formal model of semantic and syntactic regularities was devised, underlying some of the simplest types of puns (*punning riddles*). The model was then exploited in a system called JAPE that was able to automatically generate amusing puns.

Another humor-generation project was the HAHAcronym project (Stock and Strapparava, 2003), whose goal was to develop a system able to automatically generate humorous versions of existing

[6]We also like to think of this behavior as if the computer is losing its sense of humor after an overwhelming number of jokes, in a way similar to humans when they get bored and stop appreciating humor after hearing too many jokes.

acronyms, or to produce a new amusing acronym constrained to be a valid vocabulary word, starting with concepts provided by the user. The comic effect was achieved mainly by exploiting incongruity theories (e.g. finding a religious variation for a technical acronym).

Another related work, devoted this time to the problem of humor comprehension, is the study reported in (Taylor and Mazlack, 2004), focused on a very restricted type of wordplays, namely the "Knock-Knock" jokes. The goal of the study was to evaluate to what extent wordplay can be automatically identified in "Knock-Knock" jokes, and if such jokes can be reliably recognized from other non-humorous text. The algorithm was based on automatically extracted structural patterns and on heuristics heavily based on the peculiar structure of this particular type of jokes. While the wordplay recognition gave satisfactory results, the identification of jokes containing such wordplays turned out to be significantly more difficult.

6 Conclusion

A conclusion is simply the place where you got tired of thinking. (anonymous one-liner)

The creative genres of natural language have been traditionally considered outside the scope of any computational modeling. In particular humor, because of its puzzling nature, has received little attention from computational linguists. However, given the importance of humor in our everyday life, and the increasing importance of computers in our work and entertainment, we believe that studies related to computational humor will become increasingly important.

In this paper, we showed that automatic classification techniques can be successfully applied to the task of humor-recognition. Experimental results obtained on very large data sets showed that computational approaches can be efficiently used to distinguish between humorous and non-humorous texts, with significant improvements observed over apriori known baselines. To our knowledge, this is the first result of this kind reported in the literature, as we are not aware of any previous work investigating the interaction between humor and techniques for automatic classification.

Finally, through the analysis of learning curves plotting the classification performance with respect to data size, we showed that the accuracy of the automatic humor-recognizer stops improving after a certain number of examples. Given that automatic humor-recognition is a rather understudied problem, we believe that this is an important result, as it provides insights into potentially productive directions for future work. The flattened shape of the curves toward the end of the learning process suggests that rather than focusing on gathering more data, future work should concentrate on identifying more sophisticated humor-specific features, e.g. semantic oppositions, ambiguity, and others. We plan to address these aspects in future work.

References

S. Attardo. 1994. *Linguistic Theory of Humor*. Mouton de Gruyter, Berlin.

K. Binsted and G. Ritchie. 1997. Computational rules for punning riddles. *Humor*, 10(1).

C. Bucaria. 2004. Lexical and syntactic ambiguity as a source of humor. *Humor*, 17(3).

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2001. Timbl: Tilburg memory based learner, version 4.0, reference guide. Technical report, University of Antwerp.

S. Freud. 1905. *Der Witz und Seine Beziehung zum Unbewussten*. Deutike, Vienna.

T. Joachims. 1998. Text categorization with Support Vector Machines: learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*.

D.E. Knuth. 1993. *The Stanford Graph Base: A Platform for combinatorial computing*. ACM Press.

D. Lewis, Y. Yang, T. Rose, and F. Li. 2004. RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

A. McCallum and K. Nigam. 1998. A comparison of event models for Naive Bayes text classifi cation. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*.

G. Miller. 1995. Wordnet: A lexical database. *Communication of the ACM*, 38(11):39–41.

A. Nijholt, O. Stock, A. Dix, and J. Morkes, editors. 2003. *Proceedings of CHI-2003 workshop: Humor Modeling in the Interface*, Fort Lauderdale, Florida.

W. Ruch. 2002. Computers with a personality? lessons to be learned from studies of the psychology of humor. In *Proceedings of the The April Fools Day Workshop on Computational Humour*.

O. Stock and C. Strapparava. 2003. Getting serious about the development of computational humour. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico.

O. Stock, C. Strapparava, and A. Nijholt, editors. 2002. *Proceedings of the The April Fools Day Workshop on Computational Humour*, Trento.

J. Taylor and L. Mazlack. 2004. Computationally recognizing wordplay in jokes. In *Proceedings of CogSci 2004*, Chicago.

Optimizing to Arbitrary NLP Metrics using Ensemble Selection

Art Munson, Claire Cardie, Rich Caruana
Department of Computer Science
Cornell University
Ithaca, NY 14850
{mmunson, cardie, caruana}@cs.cornell.edu

Abstract

While there have been many successful applications of machine learning methods to tasks in NLP, learning algorithms are not typically designed to optimize NLP performance metrics. This paper evaluates an ensemble selection framework designed to optimize arbitrary metrics and automate the process of algorithm selection and parameter tuning. We report the results of experiments that instantiate the framework for three NLP tasks, using six learning algorithms, a wide variety of parameterizations, and 15 performance metrics. Based on our results, we make recommendations for subsequent machine-learning-based research for natural language learning.

1 Introduction

Among the most successful natural language learning techniques for a wide variety of linguistic phenomena are supervised inductive learning algorithms for classification. Because of their capabilities for accurate, robust, and efficient linguistic knowledge acquisition, they have been employed in many natural language processing (NLP) tasks.

Unfortunately, supervised classification algorithms are typically designed to optimize accuracy (*e.g.* decision trees) or mean squared error (*e.g.* neural networks). For many NLP tasks, however, these standard performance measures are inappropriate. For example, NLP data can be highly skewed in its distribution of positive and negative examples. In these situations, another metric (perhaps F-measure or a task-specific measure) that focuses on the performance of the minority cases is more appropriate. Indeed, as the NLP field matures more consideration will be given to evaluating the performance of NLP components in context (*e.g.* Is the system easy to use by end users? Does the component respect user preferences? How well does the entire system solve the specific problem?), leading to new and complicated metrics. Optimizing machine learning algorithms to arbitrary performance metrics, however, is not easily done.

To exacerbate matters, the metric of interest might change depending on how the natural language learning (NLL) component is employed. Some applications might need components with high recall, for example; others, high precision or high F-measure or low root mean squared error. To obtain good results w.r.t. the new metric, however, a different parameterization or different algorithm altogether might be called for, requiring re-training the classifier(s) from scratch.

Caruana *et al.* (2004) have recently proposed **ensemble selection** as a technique for building an ensemble of classifiers that is optimized to an arbitrary performance metric. The approach trains a large number of classifiers using multiple algorithms and parameter settings, with the idea that at least some of the classifiers will perform well on any given performance measure. The best set of classifiers, w.r.t. the target metric, is then greedily selected. (Selecting a set of size 1 is equivalent to parameter and algorithm tuning.) Like other ensemble learning methods (*e.g.* bagging (Breiman, 1996) and boosting (Freund and Schapire, 1996)), ensemble selection has been shown to exhibit reliably better performance than any of the contributing classifiers for a number of learning tasks.

In addition, ensemble selection provides an ancillary benefit: no human expertise is required in selecting an appropriate machine learning algorithm or in choosing suitable parameter settings to get good performance. This is particularly attractive for the NLP community where researchers often rely on the same one or two algorithms and use default parameter settings for simplicity. Ensemble selection is a tool usable by non-experts to find good classifiers optimized to task-specific metrics.

This paper evaluates the utility of the ensemble selection framework for NLL. We use three NLP tasks for the empirical evaluation: noun phrase coreference resolution and two problems from sentiment analysis — identifying private state frames and the hierarchy among them. The evaluation employs six learning algorithms, a wide selection of parameterizations, 8 standard metrics, and 7 task-specific metrics. Because ensemble selection subsumes parameter and algorithm selection, we also measure the impact of parameter and algorithm tuning.

Perhaps not surprisingly, we find first that no one algorithm or parameter configuration performs the best across all tasks or across all metrics. In addition, an algorithm's "tuned" performance (*i.e.* the performance after tuning parameter settings) almost universally matches or exceeds the algorithm's default performance (*i.e.* when using default parameter settings). Out of 154 total cases, the tuned classifier outperforms the default classifier 114 times, matches performance 28 times, and underperforms 12 times. Together, these results indicate the importance of algorithm and parameter selection for comparative empirical NLL studies. In particular, our results show the danger of relying on the same one or two algorithms for all tasks. These results cast doubt on conclusions regarding differences in algorithm performance for NLL experiments that give inadequate attention to parameter selection.

The results of our experiments that use ensemble selection to optimize the ensemble to arbitrary metrics are mixed. We see reliable improvements in performance across almost all of the metrics for only two of the three tasks; for the other data set, ensemble selection tends to hurt performance (although losses are very small). Perhaps more importantly for our purposes, we find that ensemble selection provides small, but consistent gains in performance when considering only the more complex, task-specific metrics — metrics that learning algorithms would find difficult to optimize.

The rest of the paper is organized as follows. Section 2 describes the general learning framework and provides an overview of ensemble selection. We present the particular instantiation of the framework employed in our experiments in Section 3. Section 4 describes the three NLP tasks. Experimental results are given in Section 5. Related work and conclusions follow (sections 6 and 7).

2 Ensemble Selection Framework

2.1 Terminology

We use the term **model** to refer to a classifier produced by some learning algorithm using some particular set of parameters. A model's **configuration** is simply the algorithm and parameter settings used to create the classifier. A **model family** is the set of models made by varying the parameters for one machine learning algorithm. Finally, a **model library** is a collection of models trained for a given task.

2.2 Framework

Abstractly, the framework is the following:

1. Select a variety of learning algorithms.
2. For each algorithm, choose a wide range of settings for the algorithm's parameters.
3. Divide data into training, tuning, and test sets.
4. Build model library.
5. Select target metrics appropriate to problem.
6. Tune parameter settings and/or run ensemble selection algorithm for target metrics.

Building the model library consists of (a) using the training data to train models for all the learning algorithms under all desired combinations of parameter settings, and (b) applying each model to the tuning and test set instances and storing the predictions for use in step (6). Note that models are placed in the library regardless of performance, even though some models have very bad performance. Intuitively, this is because there is no way to know *a priori* which models will perform well on a given metric. Note that producing the base models is fully automatic and requires no expert tuning.

Parameter Tuning: The goal of parameter tuning is to identify the best model for the task according to each target metric. Parameter tuning is handled in the standard way: for each metric, we select the model from the model library with the highest performance on the tuning data and report its performance on the test data.

Ensemble Selection Algorithm: The ensemble selection algorithm (Caruana *et al.*, 2004) ignores model-specific details by *only using the predictions made by the models:* the ensemble makes predictions by averaging the predictions of its constituents. Advantageously, this only requires that predictions made by different models fall in the same range, and that they can be averaged in a meaningful way. Otherwise, models can take any form, including other ensemble methods (*e.g.* bagging or boosting). Conceptually, ensemble selection builds on top of the models in the library and uses their performance as a starting point from which to improve.

The basic ensemble selection algorithm is:

a. Start with an empty ensemble.
b. Add the model that results in the best performance for the current ensemble with respect to the tuning data and the target metric.
c. Repeat (b) for a large, fixed number of iterations.
d. The final ensemble is the ensemble from the best performing iteration on the tuning data for the target metric.

To prevent the algorithm from overfitting the tuning data we use two enhancements given by Caruana *et al.* (2004). First, in step (b) the same model can be selected multiple times (*i.e.* selection with replacement). Second, the ensemble is initialized with the top N models (again, with respect to the target metric on the tuning data). N is picked automatically such that removing or adding a model decreases performance on the tuning data.[1]

The main advantage to this framework is its reusability. After an instantiation of the framework exists, it is straightforward to apply it to multiple NLL tasks and to add additional metrics. Steps (1) and (2) only need to be done once, regardless of the number of tasks and metrics explored. Steps (3)-(5) need only be done once per NLL task. Importantly, the model library is created once for each task (*i.e.* each model configuration is only trained once) regardless of the number (or addition) of performance metrics. Finally, finding a classifier or ensemble optimized to a new metric (step (6)) does not require re-building the model library and is very fast compared to training the classifiers—it only requires averaging the stored predictions. For example, training the model library for our smallest data set took multiple days; ensemble selection built optimized ensembles for each metric in a few minutes.

3 Framework Instantiation

In this section we describe our instantiation of the ensemble selection framework.

Algorithms: We use bagged decision trees (Breiman, 1996), boosted decision stumps (Freund and Schapire, 1996), k-nearest neighbor, a rule learner, and support vector machines (SVM's). We use the following implementations of these algorithms, respectively: `IND` decision tree package (Buntine, 1993); `WEKA` toolkit (Witten and Frank, 2000); `TiMBL` (Daelemans *et al.*, 2000); `RIPPER` (Cohen, 1995); and `SVM`^{light} (Joachims, 1999). Additionally, we use logistic regression (LR) for coreference resolution because an implementation using the `MALLET` (McCallum, 2002) toolkit was readily available for the task. The predictions from all algorithms are scaled to the range $[0, 1]$ with values close to 1 indicating positive predictions and values close to 0 indicating negative predictions.[2]

Parameter Settings: Table 1 lists the parameter settings we vary for each algorithm. Additional domain-specific parameters are also varied for coreference resolution models (see Section 4.1). The model libraries contain models corresponding to the cross product of the various parameter settings for a given algorithm.

Standard Performance Metrics: We evaluate the framework with 8 metrics: accuracy (ACC), average precision (APR), break even point (BEP), F-measure (F1), mean cross entropy (MXE), root mean squared error (RMS), area under the ROC curve (ROC), and SAR. Caruana *et al.* (2004) define SAR as $SAR = \frac{ACC+ROC+(1-RMS)}{3}$. We also evaluate the effects of model selection with task-specific metrics. These are described in Section 4. Our F-measure places equal emphasis on precision

[1]We also experimented with the bagging improvement described by Caruana *et al.* (2004). In our experiments using bagging hurt the performance of ensemble selection.

[2]We follow Caruana *et al.* (2004) in using Platt (2000) scaling to convert the SVM predictions from the range $(-\infty, \infty)$ to the required $[0, 1]$ by fitting them to a sigmoid.

Algorithm	Parameter	Values
Bagged Trees †	tree type	bayes, **c4**, cart, cart0, id3, mml, smml
	# bags	1, 5, 10, **25**
Boosted Stumps	# iterations	2, 4, 8, …, **256**, …1024, 2048
LR ‡	gaussian gamma	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, **10**, 50
K-NN	k	**1**, 3, 5
	search algorithm	**ib1**, igtree
	similarity metric	**overlap**, modified value difference
	feature weighting	**gain ratio**, information gain, chi-squared, shared variance
Rule Learner	class learning order	unordered, **pos first**, neg first, heuristic determined order
	loss ratio	0.5, **1**, 1.5, 2, 3, 4
SVM	margin tradeoff*	10^{-7}, 10^{-6}, …, $\mathbf{10^{-2}}$, $\mathbf{10^{-1}}$, …, 10^{2}
	kernel	**linear**, rbf
	rbf gamma parm	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2

Table 1: Summary of model configurations used in experiments. The default settings for each algorithm are in bold. † Bagged trees are not used for identifying PSF's since the IND package does not support features with more than 255 values. Also, for coreference resolution the number of bags is not varied and is always 25. ‡ LR is only used for coreference resolution. * SVM^{light} determines the default margin tradeoff based on data properties. We calculate this value for each data set and use the closest setting among our configurations.

and recall (*i.e.* $\beta = 1$). Note that precision and recall are calculated *with respect to the positive class*.

Ensemble Selection: For the sentiment analysis tasks, ensemble selection iterates 150 times; for the coreference task, the algorithm iterates 200 times. This should be enough iterations, given that the model libraries contain 202, 173, and 338 models. Because computing the MUC-F1 and BCUBED metrics (see Section 4.1) is expensive, ensemble selection only iterates 50 times for these metrics.

4 Tasks

Because of space issues, we necessarily provide only brief descriptions of each NLL task. Readers are referred to the cited papers to obtain detailed descriptions.

4.1 Noun Phrase Coreference Resolution

The goal for a standard noun phrase coreference resolution system is to identify the noun phrases in a document and determine which of them refer to the same entity. Entities can be people, places, things, *etc*. The resulting partitioning of noun phrases creates reference chains with one chain per entity.

We use the same problem formulation as Soon *et al.* (2001) and Ng and Cardie (2002) — a combination of classification and clustering. Briefly, every noun phrase is paired with all preceding noun phrases, creating multiple pairs. For the training data, the pairs are labeled as coreferent or not. A binary classifier is trained to predict the pair labels. During classification, the predicted labels are used to form clusters. Two noun phrases A and B share a cluster if they are either predicted as coreferent by the classifier or if they are transitively predicted as coreferent through one or more other noun phrases. Instance selection (Soon *et al.*, 2001; Ng, 2004) is used to increase the percentage of positive instances in the training set.[3]

We use the learning features described by Ng and Cardie (2002). All learning algorithms are trained with the full set of features. Additionally, the rule learner, SVM, and LR are also trained with a hand-selected subset of the features that Ng and Cardie (2002) find to outperform the full feature set. Essentially this is an additional parameter to set for the learning task.

Special Metrics: Rather than focusing on performance at the pairwise coreference classification level, performance for this task is typically reported using either the MUC metric (Vilain *et al.*, 1995) or the BCUBED metric (Bagga and Baldwin, 1998). Both of these metrics measure the degree that predicted coreference chains agree with an answer key. In particular they measure the chain-level precision and recall (and the corresponding F-measure). We abbreviate these metrics MUC-F1, and B3F1.

Data Set: For our experiments we use the MUC-6 corpus, which contains 60 documents annotated with coreference information. The training, tuning, and test sets consist of documents 1-20, 21-30, and

[3]`Soon-1` instance selection is used for all algorithms; we also use `soon-2` (Ng, 2004) instance selection for the rule learner.

31-60, respectively.

4.2 Identifying Private State Frames

NLP research has recently started looking at how to detect and understand subjectivity in discourse. A key step in this direction is automatically identifying phrases that express subjectivity. In this setting, subjectivity is defined to include implicit and explicit opinions, internal thoughts and emotions, and bias introduced through second-hand reporting. Phrases expressing any of these are called **private state frames**, which we will abbreviate as PSF.

We build directly on experiments done by Wiebe *et al.* (2003). The task is to learn to identify explicit single-word PSF's in context. One learning instance is created for every word in the corpus. Classification labels each instance as a PSF or not. We use the same features as Wiebe *et al.*

Special Metrics: Because the data is highly skewed (2% positive instances), performance measures that focus on how well the minority class is learned are of primary interest. The F-measure defined in Section 3 is a natural choice. We also evaluate performance using geometric accuracy, defined as $GACC = \sqrt{posacc \times negacc}$ where $posacc$ and $negacc$ are the accuracy with respect to the positive and negative instances (Kubat and Matwin, 1997).

Conceivably, an automatic PSF extractor with high precision and mediocre recall could be used to automate the annotation process. For this reason we measure the performance with an unbalanced F-measure that emphasizes precision. Specifically, we try $\beta = 0.5$ (F0.5) and $\beta = 0.2$ (F0.2).

Data Set: We use 400 documents from the MPQA corpus (2002), a collection of news stories manually annotated with PSF information. The 400 documents are randomly split to get 320 training, 40 tuning, and 40 testing documents.

4.3 Determining PSF Hierarchy

The third task is taken from Breck and Cardie (2004). Explicit PSF's each have a **source** that corresponds to the person or entity expressing the subjectivity. In the presence of second-hand reporting, sources are often nested. This has the effect of filtering subjectivity through a chain of sources.

Given sentences annotated with PSF information (*i.e.* which spans are PSF's), the task is to discover the hierarchy among the PSF's that corresponds to the nesting of their respective sources. From each sentence, multiple instances are created by pairing every PSF with every other PSF in the sentence.[4] Let $(PSF_{parent}, PSF_{target})$ denote one of these instances. The classification task is to decide if PSF_{parent} is the parent of PSF_{target} in the hierarchy and to associate a confidence with that prediction. The complete hierarchy can easily be constructed from the predictions by choosing for each PSF its most confidently predicted parent.

Special Metrics: Breck and Cardie (2004) measure task performance with three metrics. The first is the accuracy of the predictions over the instances. The second is a derivative of a measure used to score dependency parses. Essentially, a sentence's score is the fraction of parent links correctly identified. The score for a set of sentences is the average of the individual sentence scores. We refer to this measure as average sentence accuracy (SENTACC). The third measure is the percentage of sentences whose hierarchical structures are perfectly determined (PERFSENT).

Data Set: We use the same data set and features as Breck and Cardie (2004). The annotated sentences from 469 documents in the MPQA Corpus (MPQA Corpus, 2002) are randomly split into training (80%), tuning (10%), and test (10%) sets.

5 Experiments and Results

We evaluate the potential benefits of the ensemble selection framework with two experiments. The first experiment measures the performance improvement yielded by parameter tuning and finds the best performing algorithm. The second experiment measures the performance improvement from ensemble selection.

Performance improvements are measured in terms of performance **gain**. Let a and b be the measured performances for two models A and B on some metric. A's gain over B is simply $a - b$. A performed worse than B if the gain is negative.[5]

[4]Sentences containing fewer than two PSF's are discarded and not used.

[5]MXE and RMS have inverted scales where the best performance is 0. Gain for these metrics equals $(1 - a) - (1 - b)$ so that positive gains are always good. Similarly, where raw MXE and RMS scores are reported we show $1 - score$.

	Metric	**BAG**	**Parm∆**	**BST**	**Parm∆**	**LR**	**Parm∆**	**KNN**	**Parm∆**	**RULE**	**Parm∆**	**SVM**	**Parm∆**	**Avg ∆**
coreference	ACC	0.9861	-0.0000	*0.9861*	-0.0001	0.9849	0.0006	0.9724	0.0131	0.9840	**0.0023**	0.9859	-0.0001	0.0026
	APR	*0.5373*	**0.0000**	0.5475	0.0000	0.3195	-0.0004	0.1917	0.2843	0.2491	0.1127	0.5046	0.0323	0.0715
	BEP	*0.6010*	0.0000	0.5577	0.0193	0.3747	-0.0022	0.3243	0.2057	0.3771	0.1862	0.5965	**0.0045**	0.0689
	F1	*0.5231*	**0.0664**	0.3881	0.0000	0.4600	0.0087	0.4105	0.1383	0.4453	0.1407	0.3527	0.0571	0.0685
	MXE	0.9433	**0.0082**	*0.9373*	0.0000	0.5400	0.1828	0.4953	0.3734	0.9128	0.0222	0.9366	0.0077	0.0990
	RMS	0.8925	0.0041	*0.8882*	0.0000	0.6288	0.1278	0.8334	0.0559	0.8756	0.0097	0.8859	**0.0047**	0.0337
	ROC	0.9258	**0.0158**	*0.9466*	0.0000	0.4275	0.0022	0.7746	0.0954	0.6845	0.1990	0.8418	0.0551	0.0612
	SAR	0.9255	**0.0069**	*0.9309*	0.0000	0.6736	-0.0037	0.8515	0.0538	0.8396	0.0695	0.8955	0.0165	0.0238
	MUC-F1	*0.6691*	**0.0000**	0.6242	0.0046	0.6405	0.0344	0.5340	0.1185	0.6500	0.0291	0.5181	0.1216	0.0514
	B3F1	*0.4625*	0.0000	0.4512	0.0000	0.4423	0.0425	0.0965	0.3357	0.4249	**0.0675**	0.3323	0.1430	0.0981
PSF identification	ACC	—	—	0.9854	0.0007	—	—	0.9873	0.0011	0.9862	0.0003	*0.9886*	**0.0000**	0.0005
	APR	—	—	0.6430	0.0316	—	—	0.5588	0.1948	0.4335	0.0381	*0.7697*	**0.0372**	0.0754
	BEP	—	—	0.5954	0.0165	—	—	0.6727	0.0302	0.4436	0.0718	*0.6961*	**0.0385**	0.0393
	F1	—	—	0.5643	0.0276	—	—	0.6837	0.0019	0.5770	0.0367	*0.6741*	**0.0062**	0.0181
	MXE	—	—	0.9342	0.0029	—	—	0.8089	0.1425	0.9265	0.0062	*0.9572*	**0.0093**	0.0402
	RMS	—	—	0.8838	0.0028	—	—	0.8896	0.0118	0.8839	0.0020	*0.9000*	**0.0068**	0.0058
	ROC	—	—	0.9576	0.0121	—	—	0.8566	0.1149	0.7181	0.1593	*0.9659*	**0.0188**	0.0763
	SAR	—	—	0.9329	0.0052	—	—	0.9021	0.0407	0.8541	0.0532	*0.9420*	**0.0085**	0.0269
	GACC	—	—	0.6607	-0.0004	—	—	*0.7962*	**0.0223**	0.6610	0.0506	0.7401	0.0209	0.0233
	F0.5	—	—	0.6829	0.0221	—	—	0.7150	0.0503	0.7132	0.0000	*0.7811*	**-0.0054**	0.0167
	F0.2	—	—	0.7701	0.0157	—	—	0.7331	0.0875	*0.8171*	**0.0110**	0.8542	0.0045	0.0297
PSF hierarchy	ACC	*0.8133*	**0.0000**	0.7554	0.0009	—	—	0.7940	0.0000	0.7446	0.0428	0.7761	0.0381	0.0164
	APR	0.8166	**0.0296**	0.7455	0.0013	—	—	*0.8035*	0.0000	0.5957	0.1996	0.6363	0.1520	0.0765
	BEP	*0.7385*	**-0.0066**	0.6597	-0.0030	—	—	0.7096	0.0000	0.6317	0.0567	0.6940	0.0432	0.0181
	F1	*0.7286*	**0.0033**	0.6810	0.0226	—	—	0.7000	0.0000	0.6774	0.0525	0.6933	0.0400	0.0237
	MXE	*0.6091*	**0.0166**	0.4940	0.0076	—	—	0.0379	0.4715	0.4022	0.1197	0.4681	0.1012	0.1433
	RMS	*0.6475*	**0.0054**	0.5910	0.0033	—	—	0.6057	0.0000	0.5556	0.0514	0.5836	0.0423	0.0205
	ROC	*0.8923*	**0.0096**	0.8510	0.0000	—	—	0.8519	0.0364	0.7514	0.1094	0.7968	0.0757	0.0462
	SAR	*0.7765*	**0.0073**	0.7251	0.0009	—	—	0.7430	0.0000	0.6770	0.0672	0.7116	0.0482	0.0247
	SENTACC	*0.7571*	**0.0045**	0.7307	-0.0011	—	—	0.7399	-0.0007	0.6801	0.0141	0.6889	0.0726	0.0179
	PERFSENT	*0.4948*	**0.0069**	0.4880	0.0000	—	—	0.4880	-0.0034	0.4055	0.0206	0.4158	0.1031	0.0254

Table 2: Performance gains from parameter tuning. The left column for each algorithm family is the algorithm's performance with default parameter settings. The adjacent 'Parm∆' column gives the performance gain from tuning parameters. For each metric, the best default and tuned performance across all algorithms are *italicized* and **bold-faced**, respectively.

5.1 Experiment 1: Parameter Tuning

Experiment 1 measures, for each of the 3 tasks, the performance of every model on both the tuning and test data for every metric of interest. *Based on tuning set performance,* the best default model, the best model overall, and the best model within each family are selected. The **best default model** is the highest-scoring model that emerges after comparing algorithms without doing any parameter tuning. The **best overall model** corresponds to "tuning" both the algorithm and parameters. The **best model in each family** corresponds to "tuning" the parameters for that algorithm. *Using the test set performances,* the best family models are compared to the corresponding default models to find the gains from parameter tuning.

Table 2 lists the gains achieved by parameter tuning. Each algorithm column compares the algorithm's best model to its default model. On the coreference task, for example, the best KNN model with respect to BEP shows a 20% improvement (or gain) over the default KNN model (for a final BEP score of .5300). The "Avg ∆" column shows the average gain from parameter tuning for each metric.

For each metric, the best default model is italicized while the best overall model is bold-faced. Referring again to the coreference BEP row, the best overall model is a SVM while the best default model is a bagged decision tree. Recall that these distinctions are based on *absolute performance* and not gain. Thus, the best tuned SVM outperforms all other models on this task and metric.[6]

Three conclusions can be drawn from Table 2. *First, no algorithm performs the best on all tasks or on all metrics.* For coreference, the best overall model is either a bagged tree, a rule learner, or a SVM, depending on the target metric. Similarly, for PSF identification the best model depends on the metric, ranging from a KNN to a SVM. Interestingly, bagged decision trees on the PSF hierarchy data outperform the other algorithms on all metrics and seem especially well-suited to the task.

Second, an algorithm's best-tuned model reliably yields non-trivial gains over the corresponding default model. This trend appears to hold regardless of algorithm, metric, and data set. In 114 of the 154

[6]Another interesting example is the best overall model for BEP on the PSF hierarchy task. The baseline (a bagged tree) outperforms the "best" model (a different bagged tree) on the test set even though the best model performed better on the tuning set—otherwise it would not have been selected as the best.

cases parameter tuning improves an algorithm's performance by more than 0.001 (0.1%). In the remaining 40 cases, parameter tuning only hurts 12 times, and never by more than 0.01.

Third, the best default algorithm is not necessarily the best algorithm after tuning parameters. The coreference task, in particular, illustrates the potential problem with using default parameter settings when judging which algorithm is most suited for a problem: 7 out of 10 times the best algorithm changes after parameter tuning.

These results corroborate those found elsewhere (Daelemans and Hoste, 2002; Hoste *et al.*, 2002; Hoste, 2005)—parameter settings greatly influence performance. Further, algorithmic performance differences can change when parameters are changed. Going beyond previous work, our results also underline the need to consider multiple algorithms for NLL. Ultimately, it is important for researchers to thoroughly explore options for **both** algorithm and parameter tuning and to report these in their results.

5.2 Experiment 2: Ensemble Selection

In experiment 2 an ensemble is built to optimize each target metric. The ensemble's performance is compared to that of the best overall model for the metric. Both the ensemble and the best model are selected according to tuning set performance.

Table 3 lists the gains from ensemble selection over the best parameter tuned model. For comparison, the best default and overall performances from Table 2 are reprinted. For example, the ensemble optimized for F1 on the coreference data outperforms the best bagged tree model by about 1% (and the best default model by almost 8%).

Disappointingly, ensemble selection does not consistently improve performance. Indeed, for the PSF hierarchy task ensemble selection reliably hurts performance a small amount. For the other two tasks ensemble selection reliably improves all metrics except GACC (a small loss). In other experiments, however, we optimized F-measure with $\beta = 1.5$ for the PSF identification task. Ensemble selection hurt F1.5 by almost 2%, leading us to question the technique's reliability for our second data set. Interestingly, the aggregate metrics—metrics that measure performance by combining multiple predictions—all benefit from ensemble selection, even for the hierarchy task, *albeit* for small amounts. For our tasks these comprise a subset of the task-specific performance measures: B3F1, MUC-F1, SENTACC, and PERFSENT.

	Metric	Best Default	Best TunedΔ	Ens. Sel.Δ
coreference	ACC	0.9861	0.0002	**0.0001**
	APR	0.5373	0.0000	**0.0736**
	BEP	0.6010	0.0000	**0.0124**
	F1	0.5231	0.0664	**0.0115**
	MXE	0.9373	0.0142	**0.0035**
	RMS	0.8882	0.0023	**0.0049**
	ROC	0.9466	-0.0051	**0.0120**
	SAR	0.9309	0.0015	**0.0032**
	MUC-F1	0.6691	0.0000	**0.0073**
	B3F1	0.4625	0.0299	**0.0077**
PSF identification	ACC	0.9886	0.0000	**0.0003**
	APR	0.7697	0.0372	**0.0109**
	BEP	0.6961	0.0385	**0.0136**
	F1	0.6741	0.0062	**0.0222**
	MXE	0.9572	0.0093	**0.0029**
	RMS	0.9000	0.0068	**0.0025**
	ROC	0.9659	0.0188	**0.0043**
	SAR	0.9420	0.0085	**0.0021**
	GACC	0.7962	**0.0223**	-0.0012
	F0.5	0.7811	-0.0054	**0.0063**
	F0.2	0.8171	0.0110	**0.0803**
PSF hierarchy	ACC	**0.8133**	**0.0000**	-0.0028
	APR	0.8035	**0.0427**	-0.0064
	BEP	**0.7385**	-0.0066	0.0056
	F1	0.7286	**0.0033**	-0.0016
	MXE	0.6091	**0.0166**	-0.0012
	RMS	0.6475	**0.0054**	-0.0019
	ROC	0.8923	**0.0096**	-0.0036
	SAR	0.7765	**0.0073**	-0.0015
	SENTACC	0.7571	0.0045	**0.0024**
	PERFSENT	0.4948	0.0069	**0.0172**

Table 3: Impact from tuning and ensemble selection. *Best default* shows the performance of the best classifier with no parameter tuning (*i.e.* algorithm tuning only). *Best tuned*Δ gives the performance gain from parameter and algorithm tuning. *Ens. Sel.*Δ is the performance gain from ensemble selection over the best tuned model. The best performance for each metric is marked in bold.

While we are not surprised that the positive gains are small,[7] we are surprised at how often ensemble selection hurts performance. As a result, we investigated some of the metrics where ensemble selection hurts performance and found that ensemble selection overfits the tuning data. At this time we are not sure why this overfitting happens for these tasks and not for the ones used by Caruana *et al*. Preliminary investigations suggest that having a smaller model library is a contributing factor (Caruana *et al.* use libraries containing $\sim$ 2000 models). This is consistent with the fact that the task with the largest model library, coreference, benefits the most from ensemble selection. Perhaps the reason that ensemble selection consistently improves performance for

[7]Caruana *et al.* (2004) find the benefit from ensemble selection is only half as large as the benefit from carefully optimizing and selecting the best models in the first place.

the aggregate metrics is that these metrics are harder to overfit.

Based on our results, ensemble selection seems too unreliable for general use in NLL—at least until the model library requirements are better understood. However, ensemble selection is perhaps trustworthy enough to optimize metrics that are difficult to overfit and could not be easily optimized otherwise — in our case, the task-specific aggregate performance measures.

6 Related Work

Hoste *et al.* (2002) and Hoste (2005) study the impact of tuning parameters for k-NN and a rule-learning algorithm on word sense disambiguation and coreference resolution, respectively, and find that parameter settings greatly change results. Similar work by Daelemans and Hoste (2002) shows the fallacy of comparing algorithm performance without first tuning parameters. They find that the best algorithm for a task frequently changes after optimizing parameters. In contrast to our work, these earlier experiments investigate at most two algorithms and only measure performance with one metric per task.

7 Conclusion

We evaluate an ensemble selection framework that enables optimizing classifier performance to arbitrary performance metrics without re-training. An important side benefit of the framework is the fully automatic production of base-level models, removing the need for human expertise in choosing algorithms and parameter settings.

Our experiments show that ensemble selection, compared to simple algorithm and parameter tuning, reliably improves performance for six of the seven task-specific metrics and all four "aggregate" metrics, but only benefits *all* of the metrics for one of our three data sets. We also find that exploring multiple algorithms with a variety of settings is important for getting the best performance. Tuning parameter settings results in 0.05% to 14% average improvements, with most improvements falling between 2% and 10%. To this end, the ensemble selection *framework* can be used as an environment for automatically choosing the best algorithm and parameter settings for a given NLP classification task. More work is needed to understand when ensemble selection can be safely used for NLL.

Acknowledgements We thank Alex Niculescu-Mizil for answering questions about ensemble selection and Eric Breck for sharing his expertise with both PSF data sets. We also thank the anonymous reviewers and the Cornell NLP group for helpful suggestions on paper drafts. This work was supported by the Advanced Research and Development Activity (ARDA), by NSF Grant IIS-0208028, and by NSF Grant IIS-0412930.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Linguistic Coreference Workshop at LREC 1998*, pages 563–566, May.

Eric Breck and Claire Cardie. 2004. Playing the telephone game: Determining the hierarchical structure of perspective and speech expressions. In *COLING 2004*, pages 120–126.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Wray Buntine. 1993. Learning classification trees. In D. J. Hand, editor, *Artificial Intelligence Frontiers in Statistics*, pages 182–201. Chapman & Hall, London.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *ICML*.

William W. Cohen. 1995. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *ICML*, pages 115–123, Tahoe City, CA, July 9–12. Morgan Kaufmann.

Walter Daelemans and Véronique Hoste. 2002. Evaluation of machine learning methods for natural language processing tasks. In *LREC 2002*, pages 755–760.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2000. TiMBL: Tilburg memory based learner, version 3.0, reference guide. ILK Technical Report 00-01, Tilburg University. Available from `http://ilk.kub.nl/˜ilk/papers/ ilk0001.ps.gz`.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *ICML*, pages 148–156.

Véronique Hoste, Iris Hendrickx, Walter Daelemans, and Antal van den Bosch. 2002. Parameter optimization for machine learning of word sense disambiguation. *Natural Language Engineering*, 8(4):311–325.

Véronique Hoste. 2005. *Optimization Issues in Machine Learning of Coreference Resolution*. Ph.D. thesis, University of Antwerp.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, USA.

Miroslav Kubat and Stan Matwin. 1997. Addressing the curse of imbalanced training sets: One-sided selection. In *ICML*, pages 179–186, San Francisco, CA. Morgan Kaufmann.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

MPQA Corpus. 2002. NRRC MPQA corpus. Available from `http://nrrc.mitre.org/NRRC/Docs_Data/MPQA_04/approval_time.htm`.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*, pages 104–111.

Vincent Ng. 2004. *Improving Machine Learning Approaches to Noun Phrase Coreference Resolution*. Ph.D. thesis, Cornell University.

John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In Alexander J. Smola, Peter J. Bartlett, Bernhard Schoelköpf, and Dale Schuurmans, editors, *Advances in Large-Margin Classifiers*, pages 61–74. MIT Press.

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

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of the 6th Message Understanding Conference*, pages 45–52. Morgan Kaufmann.

Janyce Wiebe, Eric Breck, Chris Buckley, Claire Cardie, Paul Davis, Bruce Fraser, Diane Litman, David Pierce, Ellen Riloff, Theresa Wilson, David Day, and Mark Maybury. 2003. Recognizing and organizing opinions expressed in the world press. In *Papers from the AAAI Spring Symposium on New Directions in Question Answering (AAAI tech report SS-03-07)*. March 24-26, 2003. Stanford University, Palo Alto, California.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Francisco, CA.

Word Sense Disambiguation Using Sense Examples Automatically Acquired from a Second Language

Xinglong Wang
School of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh
EH8 9LW, UK
xwang@inf.ed.ac.uk

John Carroll
Department of Informatics
University of Sussex
Falmer, Brighton
BN1 9QH, UK
johnca@sussex.ac.uk

Abstract

We present a novel almost-unsupervised approach to the task of Word Sense Disambiguation (WSD). We build sense examples automatically, using large quantities of Chinese text, and English-Chinese and Chinese-English bilingual dictionaries, taking advantage of the observation that mappings between words and meanings are often different in typologically distant languages. We train a classifier on the sense examples and test it on a gold standard English WSD dataset. The evaluation gives results that exceed previous state-of-the-art results for comparable systems. We also demonstrate that a little manual effort can improve the quality of sense examples, as measured by WSD accuracy. The performance of the classifier on WSD also improves as the number of training sense examples increases.

1 Introduction

The results of the recent Senseval-3 competition (Mihalcea et al., 2004) have shown that supervised WSD methods can yield up to 72.9% accuracy[1] on words for which manually sense-tagged data are available. However, supervised methods suffer from the so-called knowledge acquisition bottleneck: they need large quantities of high quality annotated data to produce reliable results. Unfortunately, very few sense-tagged corpora are available and manual sense-tagging is extremely costly and labour intensive. One way to tackle this problem is trying to automate the sense-tagging process. For example, Agirre et al. (2001) proposed a method for building topic signatures automatically, where a topic signature is a set of words, each associated with some weight, that tend to co-occur with a certain concept. Their system queries an Internet search engine with monosemous synonyms of words that have multiple senses in WordNet (Miller et al., 1990), and then extracts topic signatures by processing text snippets returned by the search engine. They trained a classifier on the topic signatures and evaluated it on a WSD task, but the results were disappointing.

In recent years, WSD approaches that exploit differences between languages have shown great promise. Several trends are taking place simultaneously under this multilingual paradigm. A classic one is to acquire sense examples using bilingual parallel texts (Gale et al., 1992; Resnik and Yarowsky, 1997; Diab and Resnik, 2002; Ng et al., 2003): given a word-aligned parallel corpus, the different translations in a target language serve as the “sense tags” of an ambiguous word in the source language. For example, Ng et al. (2003) acquired sense examples using English-Chinese parallel corpora, which were manually or automatically aligned at sentence level and then word-aligned using software. A manual selection of target translations was then performed, grouping together senses that share the same translation in Chinese. Finally, the occurrences of the word on the English side of the parallel

[1]This figure refers to the highest accuracy achieved in the Senseval-3 English Lexical Sample task with fine-grained scoring.

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 547–554, Vancouver, October 2005. ©2005 Association for Computational Linguistics

texts were considered to have been disambiguated and "sense tagged" by the appropriate Chinese translations. A classifier was trained on the extracted sense examples and then evaluated on the nouns in Senseval-2 English Lexical Sample dataset. The results appear good numerically, but since the sense groups are not in the gold standard, comparison with other Senseval-2 results is difficult. As discussed by Ng et al., there are several problems with relying on bilingual parallel corpora for data collection. First, parallel corpora, especially accurately aligned parallel corpora are rare, although attempts have been made to mine them from the Web (Resnik, 1999). Second, it is often not possible to distinguish all senses of a word in the source language, by merely relying on parallel corpora, especially when the corpora are relatively small. This is a common problem for bilingual approaches: useful data for some words cannot be collected because different senses of polysemous words in one language often translate to the same word in the other. Using parallel corpora can aggravate this problem, because even if a word sense in the source language has a unique translation in the target language, the translation may not occur in the parallel corpora at all, due to the limited size of this resource.

To alleviate these problems, researchers seek other bilingual resources such as bilingual dictionaries, together with monolingual resources that can be obtained easily. Dagan and Itai (1994) proposed an approach to WSD using monolingual corpora, a bilingual lexicon and a parser for the source language. One of the problems of this method is that for many languages, accurate parsers do not exist. With a small amount of classified data and a large amount of unclassified data in both the source and the target languages, Li and Li (2004) proposed bilingual bootstrapping. This repeatedly constructs classifiers in the two languages in parallel and boosts the performance of the classifiers by classifying data in each of the languages and by exchanging information regarding the classified data between two languages. With a certain amount of manual work, they reported promising results, but evaluated on relatively small datasets.

In previous work, we proposed to use Chinese monolingual corpora and Chinese-English bilingual dictionaries to acquire sense examples (Wang, 2004)[2]. We evaluated the sense examples using a vector space WSD model on a small dataset containing words with binary senses, with promising results. This approach does not rely on scarce resources such as aligned parallel corpora or accurate parsers.

This paper describes further progress based on our proposal: we automatically build larger-scale sense examples and then train a Naïve Bayes classifier on them. We have evaluated our system on the English Lexical Sample Dataset from Senseval-2 and the results show conclusively that such sense examples can be used successfully in a full-scale fine-grained WSD task. We tried to analyse whether more sense examples acquired this way would improve WSD accuracy and also whether a little human effort on sense mapping could further improve WSD performance.

The reminder of the paper is organised as follows. Section 2 outlines the acquisition algorithm for sense examples. Section 3 describes details of building this resource and demonstrates our application of sense examples to WSD. We also present results and analysis in this section. Finally, we conclude in Section 4 and talk about future work.

2 Acquisition of Sense Examples

Following our previous proposal (Wang, 2004), we automatically acquire English sense examples using large quantities of Chinese text and English-Chinese and Chinese-English dictionaries. The Chinese language was chosen because it is a distant language from English and the more distant two languages are, the more likely that senses are lexicalised differently (Resnik and Yarowsky, 1999). The underlying assumption of this approach is that in general each sense of an ambiguous English word corresponds to a distinct translation in Chinese. As shown in Figure 1, firstly, the system translates senses of an English word into Chinese words, using an English-Chinese dictionary, and then retrieves text snippets from a large amount of Chinese text, with the Chinese translations as queries. Then, the Chinese text snippets are segmented and then translated back to English word by word, using a Chinese-English dic-

[2]Sense examples were referred to as "topic signatures" in that paper.

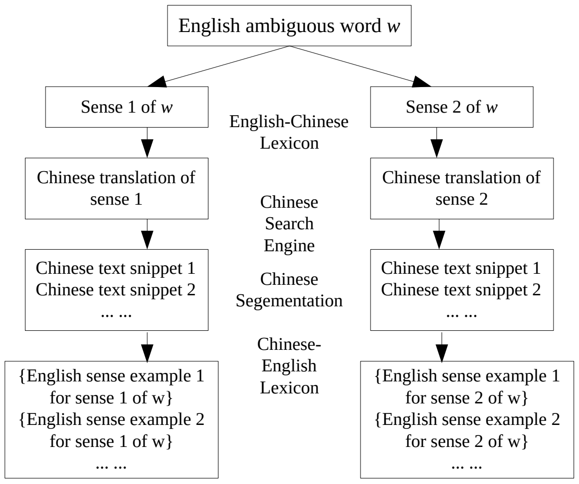

English ambiguous word *w*
Sense 1 of *w*
Sense 2 of *w*
English-Chinese Lexicon
Chinese translation of sense 1
Chinese translation of sense 2
Chinese Search Engine
Chinese text snippet 1
Chinese text snippet 2
... ...
Chinese Segementation
Chinese-English Lexicon
{English sense example 1 for sense 1 of w}
{English sense example 2 for sense 1 of w}
{English sense example 1 for sense 2 of w}
{English sense example 2 for sense 2 of w}

Figure 1. Process of automatic acquisition of sense examples. For simplicity, assume w has two senses.

tionary. In this way, for each sense, a set of sense examples is produced. As an example, suppose one wants to retrieve sense examples for the *financial* sense of *interest*. One first looks up the Chinese translations of this sense in an English-Chinese dictionary, and finds that 利息 is the right Chinese translation corresponding to this particular sense. Then, the next stage is to automatically build a collection of Chinese text snippets by either searching in a large Chinese corpus or on the Web, using 利息 as query. Since Chinese is a language written without spaces between words, one needs to use a segmentor to mark word boundaries before translating the snippets word by word back to English. The result is a collection of sense examples for the *financial* sense of *interest*, each containing a bag of words that tend to co-occur with that particular sense. For example, {*interest rate, bank, annual, economy, ...*} might be one of the sense examples extracted for the *financial* sense of *interest*. Note that words in a sense example are unordered.

Since this method acquires training data for WSD systems from raw monolingual Chinese text, it avoids the problem of the shortage of English sense-tagged corpora, and also of the shortage of aligned bilingual corpora. Also, if existing corpora are not big enough, one can always harvest more text from the Web. However, like all methods based on the cross-language translation assumption mentioned above, there are potential problems. For example, it is possible that a Chinese translation of an English sense is also ambiguous, and thus the contents of text snippets retrieved may be regarding a concept other than the one we want. In general, when the assumption does not hold, one could use the *glosses* defined in a dictionary as queries to retrieve text snippets, as comprehensive bilingual dictionaries tend to include translations to all senses of a word, where multiword translations are used when one-to-one translation is not possible. Alternatively, a human annotator could map the senses and translations by hand. As we will describe later in this paper, we chose the latter way in our experiments.

3 Experiments and Results

We firstly describe in detail how we prepared the sense examples and then describe a large scale WSD evaluation on the English Senseval-2 Lexical Sample dataset (Kilgarriff, 2001). The results show that our system trained with the sense examples achieved significantly better accuracy than comparable systems. We also show that when a little manual effort was invested in mapping the English word senses to Chinese monosemous translations, WSD performance improves accordingly. Based on further experiments on a standard binary WSD dataset, we also show that the technique scales up satisfactorily so that more sense examples help achieve better WSD accuracy.

3.1 Building Sense Examples

Following the approach described in Section 2, we built sense examples for the 44 words in the Senseval-2 dataset[3]. These 44 words have 223 senses in total to disambiguate. The first step was translating English senses to Chinese. We used the *Yahoo! Student English-Chinese On-line Dictionary*[4], as well as a more comprehensive electronic dictionary. This is because the *Yahoo!* dictionary is designed for English learners, and its sense granularity is rather coarse-grained. It is good enough for words with fewer or coarse-grained senses. How-

[3]These 44 words cover all nouns and adjectives in the Senseval-2 dataset, but exclude verbs. We discuss this point in section 3.2.

[4]See: http://cn.yahoo.com/dictionary.

ever, the Senseval-2 Lexical Sample task[5] uses WordNet 1.7 as gold standard, which has very fine sense distinctions and translation granularity in the *Yahoo!* dictionary does not conform to this standard. *PowerWord 2002*[6] was chosen as a supplementary dictionary because it integrates several comprehensive English-Chinese dictionaries in a single application. For each sense of an English word entry, both *Yahoo!* and *PowerWord 2002* dictionaries list not only Chinese translations but also English glosses, which provides a bridge between WordNet synsets and Chinese translations in the dictionaries. In detail, to automatically find a Chinese translation for sense s of an English word w, our system looks up w in both dictionaries and determines whether w has the same or greater number of senses as in WordNet. If it does, in one of the bilingual dictionaries, we locate the English gloss g which has the maximum number of overlapping words with the gloss for s in the WordNet synset. The Chinese translation associated with g is then selected. Although this simple method successfully identified Chinese translations for 23 out of the 44 words (52%), translations for the remaining word senses remain unknown because the sense distinctions are different between our bilingual dictionaries and WordNet. In fact, unless an English-Chinese bilingual WordNet becomes available, this problem is inevitable. For our experiments, we solved the problem by manually looking up dictionaries and identifying translations. For each one of the 44 words, *PowerWord 2002* provides more Chinese translations than the number of its synsets in WordNet 1.7. Thus the annotator simply selects the Chinese translations that he considers a best match to the corresponding English senses. This task took an hour for an annotator who speaks both languages fluently.

It is possible that the Chinese translations are also ambiguous, which can make the topic of a collection of text snippets deviate from what is expected. For example, the *oral* sense of *mouth* can be translated as 口 or 口腔 in Chinese. However, the first translation (口) is a single-character word and is highly ambiguous: by combining with other characters, its meaning varies. For example, 出口 means “an exit” or “to export”. On the other hand, the second translation (口腔) is monosemous and should be used. To assess the influence of such “ambiguous translations”, we carried out experiments involving more human labour to verify the translations. The same annotator manually eliminated those highly ambiguous Chinese translations and then replaced them with less ambiguous or ideally monosemous Chinese translations. This process changed roughly half of the translations and took about five hours. We compared the basic system with this manually improved one. The results are presented in section 3.2.

Using translations as queries, the sense examples were automatically extracted from *the Chinese Gigaword Corpus* (CGC), distributed by the LDC[7], which contains 2.7GB newswire text, of which 900MB are sourced from *Xinhua News Agency of Beijing*, and 1.8GB are drawn from *Central News* from Taiwan. A small percentage of words have different meanings in these two Chinese dialects, and since the Chinese-English dictionary (*LDC Mandarin-English Translation Lexicon Version 3.0*) we use later is compiled with Mandarin usages in mind, we mainly retrieve data from *Xinhua News*. We set a threshold of 100, and only when the amount of snippets retrieved from *Xinhua News* is smaller than 100, do we turn to *Central News* to collect more data. Specifically, for 48 out of the 223 (22%) Chinese queries, the system retrieved less than 100 instances from *Xinhua News* so it extracted more data from *Central News*. In theory, if the training data is still not enough, one could always turn to other text resources, such as the Web.

To decide the optimal length of text snippets to retrieve, we carried out pilot experiments with two length settings: 250 ($\approx$ 110 English words) and 400 ($\approx$ 175 English words) Chinese characters, and found that more context words helped improve WSD performance (results not shown). Therefore, we retrieve text snippets with a length of 400 characters.

We then segmented all text snippets, using an application *ICTCLAS*[8]. After the segmentor marked

[5]The task has two variations: one to disambiguate fine-grained senses and the other to coarse-grained ones. We evaluated our sense examples on the former variation, which is obviously more difficult.

[6]A commercial electronic dictionary application. We used the free on-line version at: http://cb.kingsoft.com.

[7]Available at: http://www.ldc.upenn.edu/Catalog/

[8]See: http://mtgroup.ict.ac.cn/~zhp/ICTCLAS

all word boundaries, the system automatically translated the text snippets word by word using the electronic *LDC Mandarin-English Translation Lexicon 3.0*. As expected, the lexicon does not cover all Chinese words. We simply discarded those Chinese words that do not have an entry in this lexicon. We also discarded those Chinese words with multiword English translations. Since the discarded words can be informative, one direction of our research in the future is to find an up-to-date wide coverage dictionary, and to see how much difference it will make. Finally, we filtered the sense examples with a stop-word list, to ensure only content words were included.

We ended up with 223 sets of sense examples for all senses of the 44 nouns and adjectives in the test dataset. Each sense example contains a set of words that were translated from a Chinese text snippet, whose content should closely relate to the English word sense in question. Words in a sense example are unordered, because in this work we only used bag-of-words information. Except for the very small amount of manual work described above to map WordNet glosses to those in English-Chinese dictionaries, the whole process is automatic.

3.2 WSD Experiments on Senseval-2 Lexical Sample dataset

The Senseval-2 English Lexical Sample Dataset consists of manually sense-tagged training and test instances for nouns, adjectives and verbs. We only tested our system on nouns and adjectives because verbs often have finer sense distinctions, which would mean more manual work would need to be done when mapping WordNet synsets to English-Chinese dictionary glosses. This would involve us in a rather different kind of enterprise since we would have moved from an almost-unsupervised to a more supervised setup.

We did not use the training data supplied with the dataset. Instead, we train a classifier on our automatically built sense examples and test it on the test data provided. In theory, any machine learning classifier can be applied. We chose the Naïve Bayes algorithm with kernel estimation[9] (John and Langley, 1995) which outperformed a few other classifiers in our pilot experiments on other datasets (results not shown). The average length of a sense example is 35 words, which is much shorter than the length of the text snippets, which was set to 400 Chinese characters (≈ 175 English words). This is because function words and words that are not listed in the *LDC Mandarin-English* lexicon were eliminated. We did not apply any weighting to the features because performance went down in our pilot experiments when we applied a TF.IDF weighting scheme (results not shown). We also limited the maximum number of

	Sys A		**Sys B**		**Baselines & A Senseval-2 Entry**			
Word	**Basic**	**MW**	**Basic**	**MW**	**RB**	**MFB**	**Lesk(U)**	**UNED**
art-n(5)	29.9	51.0	39.8	59.6	16.3	41.8	16.3	50.0
authority-n(7)	20.7	22.8	21.5	23.7	10.0	39.1	30.4	34.8
bar-n(13)	41.1	48.3	44.7	52.0	3.3	38.4	2.0	27.8
blind-a(3)	74.5	74.5	74.5	75.0	40.0	78.2	32.7	74.5
bum-n(4)	60.0	60.0	64.4	62.2	15.6	68.9	53.3	11.1
chair-n(4)	81.2	82.6	80.0	82.9	23.2	76.8	56.5	81.2
channel-n(7)	31.5	35.6	32.4	36.5	12.3	13.7	21.9	17.8
child-n(4)	56.3	56.3	56.3	56.3	18.8	54.7	56.2	43.8
church-n(3)	53.1	56.3	59.4	59.4	29.7	56.2	45.3	62.5
circuit-n(6)	48.2	68.2	48.8	69.8	10.6	27.1	5.9	55.3
colourless-a(2)	45.7	45.7	66.7	69.4	42.9	65.7	54.3	31.4
cool-a(6)	26.9	26.9	50.9	50.9	13.5	46.2	9.6	46.2
day-n(9)	32.2	32.9	32.4	33.1	7.6	60.0	0	20.0
detention-n(2)	62.5	84.4	60.6	84.8	43.8	62.5	43.8	78.1
dyke-n(2)	85.7	85.7	82.8	86.2	28.6	53.6	57.1	35.7
facility-n(5)	20.7	22.4	27.1	28.8	13.8	48.3	46.6	25.9
faithful-a(3)	69.6	69.6	66.7	66.7	21.7	78.3	26.1	78.3
fatigue-n(4)	76.7	74.4	77.3	77.3	25.6	76.7	44.2	86.0
feeling-n(6)	11.7	11.7	50.0	50.0	9.8	56.9	2.0	60.8
fine-a(9)	8.6	11.4	34.3	32.9	7.1	42.9	5.7	44.3
fit-a(3)	44.8	44.8	44.8	44.8	31.0	58.6	3.4	48.3
free-a(8)	29.3	37.8	37.3	48.2	15.9	35.4	7.3	35.4
graceful-a(2)	58.6	58.6	70.0	73.3	62.1	79.3	72.4	79.3
green-a(7)	53.2	58.5	53.2	58.5	21.3	75.5	10.6	78.7
grip-n(7)	35.3	37.3	35.3	37.3	19.6	35.3	17.6	21.6
hearth-n(3)	46.9	50.0	48.5	51.5	31.2	71.9	81.2	65.6
holiday-n(2)	64.5	74.2	64.5	75.0	38.7	77.4	29.0	54.8
lady-n(3)	69.2	73.6	71.7	77.8	28.3	64.2	50.9	58.5
local-a(3)	36.8	42.1	38.5	43.6	26.3	55.3	31.6	34.2
material-n(5)	39.1	44.9	47.8	49.3	10.1	20.3	44.9	53.6
mouth-n(8)	38.3	41.7	38.3	41.7	11.7	36.7	31.7	48.3
nation-n(3)	35.1	35.1	39.5	39.5	21.6	78.4	18.9	70.3
natural-a(10)	14.6	32.0	14.6	34.0	6.8	27.2	6.8	44.7
nature-n(5)	23.9	26.1	27.7	31.9	15.2	45.7	41.3	23.9
oblique-a(2)	72.4	72.4	72.4	73.3	44.8	69.0	72.4	27.6
post-n(8)	34.7	45.6	34.7	45.6	10.1	31.6	6.3	41.8
restraint-n(6)	6.7	6.7	17.4	19.6	11.1	28.9	28.9	17.8
sense-n(5)	20.7	43.4	25.9	46.3	24.5	24.5	24.5	30.2
simple-a(7)	45.5	45.5	49.3	50.7	13.6	51.5	12.1	51.5
solemn-a(2)	64.0	76.0	73.1	76.9	32.0	96.0	24.0	96.0
spade-n(3)	66.7	63.6	67.6	70.6	18.2	63.6	60.6	54.5
stress-n(5)	37.5	37.5	45.0	45.0	12.8	48.7	2.6	20.5
vital-a(4)	42.1	42.1	41.0	46.2	21.1	92.1	0	94.7
yew-n(2)	21.4	25.0	82.8	89.7	57.1	78.6	17.9	71.4
Avg.	**40.7**	**46.0**	**46.1**	**<u>52.0</u>**	**18.1**	**50.5**	**24.6**	**46.4**

Table 1. WSD accuracy on words in the English Senseval-2 Lexical Sample dataset. The left most column shows words, their POS tags and how many senses they have. "Sys A" and "Sys B" are our systems, and "MW" denotes a multi-word detection module was used in conjunction with the "Basic" system. For comparison, it also shows two baselines: "RB" is the random baseline and "MFB" is the most-frequent-sense baseline. "UNED" is one of the best unsupervised participants in the Senseval-2 competition and "Lesk(U)" is the highest unsupervised-baseline set in the workshop. All accuracies are expressed as percentages.

[9]We used the implementation in the Weka machine learning package, available at: http://www.cs.waikato.ac.nz/~ml/weka.

training sense examples to 6000, for efficiency purposes. We attempted to tag every test data instance, so our coverage (on nouns and adjectives) is 100%.

To assess the influence of ambiguous Chinese translations, we prepared two sets of training data. As described in section 3.1: sense examples in the first set were prepared without taking ambiguity in Chinese text into consideration, while those in the second set were prepared with a little more human effort involved trying to reduce ambiguity by using less ambiguous translations. We call the system trained on the first set "Sys A" and the one trained on the second "Sys B".

In this lexical sample task, multiwords are expected to be picked out by participating WSD systems. For example, the answer *art collection* should be supplied when this multiword occurs in a test instance. It would be judged wrong if one tagged the *art* in *art collection* as the *artworks* sense, even though one could argue that this was also a correct answer. To deal with multiwords, we implemented a very simple detection module, which tries to match multiword entries in WordNet to the ambiguous word and its left and right neighbours. For example, if the module finds *art collection* is an entry in WordNet, it tags all occurrences of this multiword in the test data, regardless of the prediction by the classifier.

The results are shown in Table 1. Our "Sys B" system, with and without the multiword detection module, outperformed "Sys A", which shows that sense examples acquired with less ambiguous Chinese translations contain less noise and therefore boost WSD performance. For comparison, the table also shows various baseline performance figures and a system that participated in Senseval-2[10]. Considering that the manual work involved in our approach is negligible compared with manual sense-tagging, we classify our systems as unsupervised and we should aim to beat the random baseline. This all four of our systems do easily. We also easily beat another unsupervised baseline – the Lesk (1986) baseline, which disambiguates words using WordNet definitions. The MFB baseline is actually a 'supervised' baseline, since an unsupervised system does not have such prior knowledge beforehand. McCarthy et al. (2004) argue that this is a very tough baseline for an unsupervised WSD system to beat. Our "Sys B" with multiword detection exceeds it. "Sys B" also exceeds the performance of UNED (Fernández-Amorós et al., 2001), which was the second-best ranked[11] unsupervised systems in the Senseval-2 competition.

There are a number of factors that can influence WSD performance. The distribution of training data for senses is one. In our experiments, we used all sense examples that we built for a sense (with an upper bound of 6000). However, the distribution of senses in English text often does not match the distribution of their corresponding Chinese translations in Chinese text. For example, suppose an English word w has two senses: s_1 and s_2, where s_1 rarely occurs in English text, whereas sense s_2 is used frequently. Also suppose s_1's Chinese translation is much more frequently used than s_2's translation in Chinese text. Thus, the distribution of the two senses in English is different from that of the translations in Chinese. As a result, the numbers of sense examples we would acquire for the two senses would be distributed as if they were in Chinese text. A classifier trained on this data would then tend to predict unseen test instances in favour of the wrong distribution. The word *nation*, for example, has three senses, of which the *country* sense is used more frequently in English. However, in Chinese, the *country* sense and the *people* sense are almost equally distributed, which might be the reason for its WSD accuracy being lower with our systems than most of the other words. A possible way to alleviate this problem is to select training sense examples according to an estimated distribution in natural English text, which can be done by analysing available sense-tagged corpora with help of smoothing techniques, or with the unsupervised approach of (McCarthy et al., 2004).

Cultural differences can cause difficulty in retrieving sufficient training data. For example, translations of senses of *church* and *hearth* appear only infrequently in Chinese text. Thus, it is hard to build sense examples for these words. Another problem,

[10]Accuracies for each word and averages were calculated by us, based on the information on Senseval-2 Website. See: http://www.sle.sharp.co.uk/senseval2/.

[11]One system performed better but their answers were not on the official Senseval-2 website so that we could not do the comparison. Also, that system did not attempt to disambiguate as many words as UNED and us.

as mentioned above, is that translations of English senses can be ambiguous in Chinese. For example, Chinese translations of the words *vital*, *natural*, *local* etc. are also ambiguous to some extent, and this might be a reason for their low performance. One way to solve this, as we described, is to manually check the translations. Another automatic way is that, before retrieving text snippets, we could segment or even parse the Chinese corpora, which should reduce the level of ambiguity and lead to better sense examples.

3.3 Further WSD Experiments

One of the strengths of our approach is that training data come cheaply and relatively easily. However, the sense examples are acquired automatically and they inevitably contain a certain amount of noise, which may cause problems for the classifier. To assess the relationship between accuracy and the size of training data, we carried out a series of experiments, feeding the classifier with different numbers of sense examples as training data.

For these experiments, we used another standard WSD dataset, the TWA dataset. This is a manually sense-tagged corpus (Mihalcea, 2003), which contains 2-way sense-tagged text instances, drawn from the British National Corpus, for 6 nouns. We first built sense examples for all the 12 senses using the approach described above, then trained the same Naïve Bayes algorithm (NB) on different numbers of sense examples.

In detail, for all of the 6 words, we did the following: given a word w_i, we randomly selected n sense examples for each of its senses s_i, from the total amount of sense examples built for s_i. Then the NB algorithm was trained on the $2 * n$ examples and tested on w_i's test instances in TWA. We recorded the accuracy and repeated this process 200 times and calculated the mean and variance of the 200 accuracies. Then we assigned another value to n and iterated the above process until n took all the predefined values. In our experiments, n was taken from {*50, 100, 150, 200, 400, 600, 800, 1000, 1200*} for words *motion*, *plant* and *tank* and from {*50, 100, 150, 200, 250, 300, 350*} for *bass*, *crane* and *palm*, because there were less sense example data available for the latter three words. Finally, we used the t-test ($p = 0.05$) on pairwise sets of means and variances to see if improvements were statistically significant.

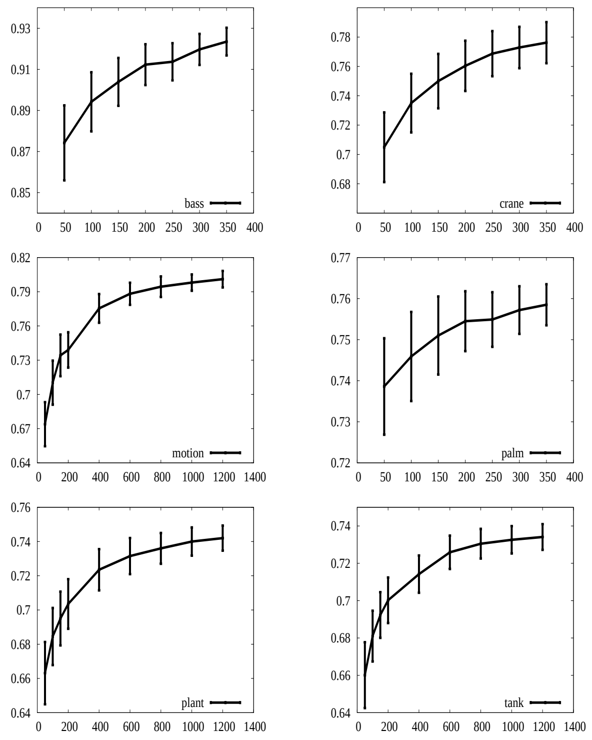


Figure 2. Accuracy scores with increasing number of training sense examples. Each bar is a standard deviation.

The results are shown in Figure 2[12]. 34 out of 42 t-scores are greater than the t-test critical values, so we are fairly confident that the more training sense examples used, the more accurate the NB classifier becomes on this disambiguation task.

4 Conclusions and Future Work

We have presented WSD systems that use sense examples as training data. Sense examples are acquired automatically from large quantities of Chinese text, with the help of Chinese-English and English-Chinese dictionaries. We have tested our WSD systems on the English Senseval-2 Lexical Sample dataset, and our best system outperformed comparable state-of-the-art unsupervised systems. Also, we found that increasing the number of the sense examples significantly improved WSD performance. Since sense examples can be obtained very cheaply from any large Chinese text collection, in-

[12]These experiments showed that our systems outperformed the most-frequent-sense baseline and Mihalcea's unsupervised system (2003).

cluding the Web, our approach is a way to tackle the knowledge acquisition bottleneck.

There are a number of future directions that we could investigate. Firstly, instead of using a bilingual dictionary to translate Chinese text snippets back to English, we could use machine translation software. Secondly, we could try this approach on other language pairs, Japanese-English, for example. This is also a possible solution to the problem that ambiguity may be preserved between Chinese and English. In other words, when a Chinese translation of an English sense is still ambiguous, we could try to collect sense examples using translation in a third language, Japanese, for instance. Thirdly, it would be interesting to try to tackle the problem of Chinese WSD using sense examples built using English, the reverse process to the one described in this paper.

Acknowledgements

This research was funded by EU IST-2001-34460 project MEANING: Developing Multilingual Web-Scale Language Technologies.

References

Eneko Agirre, Olatz Ansa, David Martinez, and Eduard Hovy. 2001. Enriching WordNet concepts with topic signatures. In *Proceedings of the NAACL workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*. Pittsburgh, USA.

Ido Dagan and Alon Itai. 1994. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, 20(4):563–596.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the* 40^{th} *Anniversary Meeting of the Association for Computational Linguistics (ACL-02)*. Philadelphia, USA.

David Fernández-Amorós, Julio Gonzalo, and Felisa Verdejo. 2001. The UNED systems at Senseval-2. In *Poceedings of Second International Wordshop on Evaluating Word Sense Disambiguation Systems (Senseval-2)*. Toulouse, France.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. Using bilingual materials to develop word sense disambiguation methods. In *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 101–112.

George H. John and Pat Langley. 1995. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

Adam Kilgarriff. 2001. English lexical sample task description. In *Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*. Toulouse, France.

Michael E. Lesk. 1986. Automated sense disambiguation using machine-readable dictionaries: how to tell a pinecone from an ice cream cone. In *Proceedings of the SIGDOC Conference*.

Hang Li and Cong Li. 2004. Word translation disambiguation using bilingual bootstrapping. *Computational Linguistics*, 20(4):563–596.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*. Barcelona, Spain.

Rada Mihalcea, Timothy Chklovski, and Adam Killgariff. 2004. The Senseval-3 English lexical sample task. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*.

Rada Mihalcea. 2003. The role of non-ambiguous words in natural language disambiguation. In *Proceedings of the Conference on Recent Advances in Natural Language Processing, RANLP 2003*. Borovetz, Bulgaria.

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

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: an empirical study. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Philip Resnik and David Yarowsky. 1997. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What and How?*, pages 79–86.

Philip Resnik and David Yarowsky. 1999. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5(2):113–133.

Philip Resnik. 1999. Mining the Web for bilingual text. In *Proceedings of the* 37^{th} *Annual Meeting of the Association for Computational Linguistics*.

Xinglong Wang. 2004. Automatic acquisition of English topic signatures based on a second language. In *Proceedings of the Student Research Workshop at ACL 2004*. Barcelona, Spain.

Using MONA for Querying Linguistic Treebanks

Stephan Kepser[*]
Collaborative Research Centre 441
University of Tübingen
Tübingen, Germany
kepser@sfs.uni-tuebingen.de

Abstract

MONA is an automata toolkit providing a compiler for compiling formulae of monadic second order logic on strings or trees into string automata or tree automata. In this paper, we evaluate the option of using MONA as a treebank query tool. Unfortunately, we find that MONA is not an option. There are several reasons why the main being unsustainable query answer times. If the treebank contains larger trees with more than 100 nodes, then even the processing of simple queries may take hours.

1 Introduction

In recent years large amounts of electronic texts have become available providing a new base for empirical studies in linguistics and offering a chance to linguists to compare their theories with large amounts of utterances from "the real world". While tagging with morphosyntactic categories has become a standard for almost all corpora, more and more of them are nowadays annotated with refined syntactic information. Examples are the Penn Treebank (Marcus et al., 1993) for American English annotated at the University of Pennsylvania, the French treebank (Abeillé and Clément, 1999) developed in Paris, the TIGER Corpus (Brants et al., 2002) for German annotated at the Universities of Saarbrücken and Stuttgart, and the Tübingen Treebanks (Hinrichs et al., 2000) for Japanese, German and English from the University of Tübingen. To make these rich syntactic annotations accessible for linguists, the development of powerful query tools is an obvious need and has become an important task in computational linguistics.

Consequently, a number of treebank query tools have been developed. Probably amongst the most important ones are CorpusSearch (Randall, 2000), ICECUP III (Wallis and Nelson, 2000), fsq (Kepser, 2003), TGrep2 (Rohde, 2001), and TIGERSearch (König and Lezius, 2000). A common feature of these tools is the relatively low expressive power of their query languages. Explicit or implicit references to nodes in a tree are mostly interpreted existentially. The notable exception is fsq, which employs full first order logic as its query language.

The importance of the expressive power of the query language is a consequence of the sizes of the available treebanks, which can contain several ten-thousand trees. It is clearly impossible to browse these treebanks manually searching for linguistic phenomena. But a query tool that does not permit the user to specify the sought linguistic phenomenon quite precisely is not too helpful, either. If the user can only approximate the phenomenon he seeks answer sets will be very big, often containing several hundred to thousand trees. Weeding through answer sets of this size is cumbersome and not really fruitful. If the task is to gain small answer sets, then query languages must be powerful. The reason why the above mentioned query tools still offer query languages of limited expressive power is the fear that

[*] This research was funded by a German Science Foundation grant (DFG SFB441-6).

there may be a price to be paid for offering a powerful query language, namely longer query answer times due to more complex query evaluation algorithms.

At least on a theoretical level, this fear is not necessarily justified. As was recently shown by Kepser (2004), there exists a powerful query language with a query evaluation algorithm of low complexity. The query language is monadic second-order logic (MSO henceforth), an extension of first-order logic that additionally allows for the quantification over *sets* of tree nodes. The fact that makes this language so appealing beyond its expressive power is that the evaluation time of an MSO query on a tree is only linear in the size of the tree. The query evaluation algorithm proceeds in two steps. In the first step, a query is compiled into an equivalent tree automaton. In the second, the automaton is run on each tree of the treebank. Since a run of an automaton on a tree is linear in the size of the tree, the evaluation of an MSO query is linear in the size of a tree.

There has sometimes been the question whether the expressive power of MSO is really needed. Beyond the statements above about retrieving small answer sets there is an important argument concerning the expressive power of the grammars underlying the annotation of the treebanks. A standard assumption in the description of the syntax of natural languages is that at least context-free string grammars are required. On the level of trees, these correspond to regular tree grammars (Gécseg and Steinby, 1997). It is natural to demand that the expressive power of the query language matches the expressive power of the underlying grammar. Otherwise there can be linguistic phenomena annotated in the treebank for which a user cannot directly query. The query language which exactly matches the expressive power of regular tree grammars is MSO. In other words, a set of trees is definable by a regular tree grammar iff there is an MSO formula that defines this set of trees (Gécseg and Steinby, 1997). Hence MSO is a natural choice of query language under the given assumption that the syntax of natural language is (at least) context-free on the string or token level.

Since the use of MSO as a query language for treebanks is – at least on a theoretical level – quite appealing, it is worth trying to develop a query system that brings these theoretical concepts to practise. The largest and most demanding subpart of this enterprise is the development of a tree automata toolkit, a toolkit that compiles formulae into tree automata and performs standard operations on tree automata such as union, intersection, negation, and determination. Since this task is very demanding, it makes sense to investigate whether one could use existing tree automata toolkits before starting to develop a new one. To the authors' knowledge, there exists only one of-the-shelf usable tree automata toolkit, and that is MONA (Klarlund, 1998). It is the aim of this paper to give an evaluation of using MONA for querying linguistic treebanks.

2 The Tree Automata Toolkit MONA

Tree automata are generalisations of finite state automata to trees. For a general introduction to tree automata, we refer the reader to (Gécseg and Steinby, 1997). There exists a strong connection between tree automata and MSO. A set of trees is definable by an MSO formula if and only if there exists a tree automaton accepting this set. This equivalence is constructive, there is an algorithm that constructs an automaton from a given MSO formula.

MONA is an implementation of this relationship. It is being developed by Nils Klarlund, Anders Møller, and Michael Schwartzbach. Its intended main uses are hardware and program verification. MONA is actually an implementation of the compilation of monadic second order logic on *strings and* trees into finite state automata or tree automata, respectively. But we focus exclusively on the tree part here. As we will see later, MONA was not developed with linguistic applications in mind.

2.1 The Language of MONA

The language of MONA is pure monadic second order logic of two successors. We will only mention the part of the language that is needed for describing trees. There are first-order and second-order terms. A first-order variable is a first-order term. The constant *root* is a first-order term denoting the root node of a tree. If t is a first-order term and s is a non-empty sequence of 0's and 1's, then $t.s$ is a first-order term. 0 denotes the left daughter, and 1 the right daughter of a node. A sequence of 0's and 1's denotes a path in the tree. The term *root*.011, e.g., denotes the node that is reached from the root by first going to the left daughter and then going twice down to the right daughter. A set variable is a second-order term. If $t_1,\dots,t_k$ are first-order terms then $\{t_1,\dots,t_k\}$ is a second-order term. We consider the following formulae. Let t,t' be first-order terms and T,T' be second order terms. Atomic formulae are

- $t = t'$ – Equality of nodes,
- $T = T'$ – Equality of node sets,
- t `in` T – t is a member of set T,
- `empty`(T) – Set T is empty.

Formulae are constructed from atomic formulae using the boolean connectives and quantification. Let ϕ and ψ be formulae. Then we define complex formulae as

- $\sim \phi$ – Negation of ϕ,
- ϕ & ψ – Conjunction of ϕ and ψ,
- $\phi \mid \psi$ – Disjunction of ϕ and ψ,
- ϕ `=>` ψ – Implication of ϕ and ψ,
- `ex1` $x : \phi$ – First-order existential quantification of x in ϕ,
- `all1` $x : \phi$ – First-order universal quantification of x in ϕ,
- `ex2` $X : \phi$ – Existential quantification of set X in ϕ,
- `all2` $X : \phi$ – Universal quantification of set X in ϕ.

We note that there is no way to extend this language. This has three important consequences. Firstly, we are restricted to using *binary* trees only. And secondly, we cannot accommodate linguistic labels in a direct way. We have to find some coding. Finally, and this is a significant drawback that may exclude the use of MONA for many applications, we cannot code the tokens, the word sequence at the leaves of a tree. Hence we can neither query for particular words or sequences of words. We can only query the structure of a tree – including the labels.

2.2 The MONA Compiler

The main program of MONA is a compiler that compiles formulae in the above described language into tree automata. The input is a file containing the formulae. The output is an analysis of the automaton that is constructed. In particular, it is stated whether the formula is satisfiable at all, i.e., whether an automaton can be constructed.

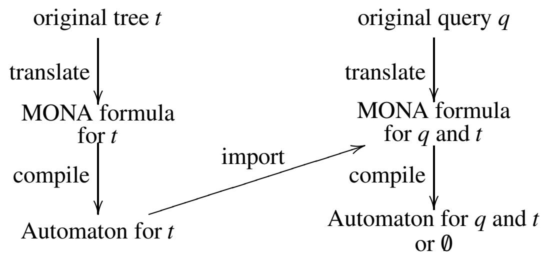


Figure 1: Method of using MONA for querying.

MONA does not provide a method to execute an automaton on a tree. But if a formula can be compiled into an automaton, this automaton can be output to file. And a file containing an automaton can be imported into a file containing a formula. We therefore use the following strategy to query treebanks using MONA. Each tree from the treebank is translated into a formula in the MONA language. How this can be done, will be described later. The formula representing the tree is then compiled into an automaton and written to file. Now the treebank exists as a set of automata files. A query to the original treebank will also be translated into a MONA formula. For each tree of the treebank, this formula is extended by an import statement for the automaton representing the tree. If and only if the extended formula representing query and tree can be compiled into an automaton, then the tree is a match for the original query. This way we can use MONA to query the treebank. The method is depicted in Figure 1.

3 The Tübingen Treebanks

In order to evaluate the usability of MONA as a query tool we had to chose some treebank to do our evaluation on. We opted for the Tübingen Treebank of spoken German. The Tübingen Treebanks, annotated at the University of Tübingen, comprise a German, an English and a Japanese treebank consisting of spoken dialogs restricted to the domain of arranging business appointments. For our evaluation, we focus on the German treebank (TüBa-D/S) (Stegmann et al., 2000; Hinrichs et al., 2000) that contains approximately 38.000 trees.

The treebank is part-of-speech tagged using the Stuttgart-Tübingen tag set (STTS) developed by

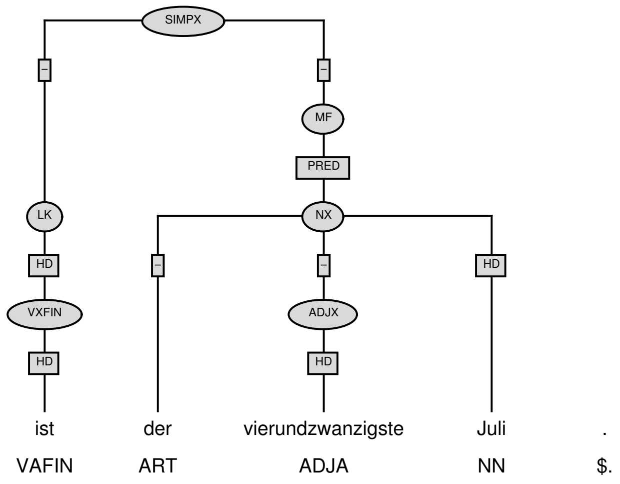


Figure 2: An example tree from TüBa-D/S.

Schiller et al. (1995). One of the design decisions for the development of the treebank was the commitment to reusability. As a consequence, the choice of the syntactic annotation scheme should not reflect a particular syntactic theory but rather be as theory-neutral as possible. Therefore a surface-oriented scheme was adopted to structure German sentences that uses the notion of topological fields in a sense similar to that of Höhle (1985). The verbal elements have the categories `LK` (*linke Klammer*) and `VC` (*verbal complex*); roughly everything preceeding the `LK` forms the "Vorfeld" `VF`, everything between `LK` and `VC` forms the "Mittelfeld" `MF`, and the material following the `VC` forms the "Nachfeld" `NF`.

The treebank is annotated with syntactic categories as node labels, grammatical functions as edge labels and dependency relations. The syntactic categories follow traditional phrase structure and the theory of topological fields. An example of a tree can be found in Figure 2. To cope with the characteristics of spontaneous speech, the data structures in the Tübingen Treebanks are of a more general form than trees. For example, an entry may consist of several tree structures. It may also contain completely disconnected nodes. In contrast to TIGER or the Penn Treebank, there are neither crossing branches nor empty categories.

There is no particular reason why we chose this treebank. Many others could have been used as well for testing the applicability of MONA.

4 Converting Trees into Automata

4.1 Translating Trees into Tree Descriptions

When translating trees from the treebank into MONA formulae describing these trees we consider proper trees only. Many treebanks, including TüBa-D/S, contain more complex structures than proper trees. For the evaluation purpose here we simplify these structures as follows. We ignore the secondary relations. And we introduce a new super root. All disconnected subparts are connected to this super root. Note that we employ this restriction for the evaluation purpose only. The general method does not require these restrictions, because even more complex tree-like structures can be recoded into proper binary trees, as is shown in (Kepser, 2004).

As stated above, the translation of trees into formulae has to perform two tasks. The trees, which are arbitrarily branching, must be transformed into binary trees. And the linguistic labels, i.e., the node categories and grammatical functions, have to be coded. For the transformation into binary trees, we employ the First-Daughter-Next-Sibling encoding, a rather standard technique. Consider an arbitrary node x in the tree. If x has any daughters, its leftmost daughter will become the left daughter of x in the transformed tree. If x has any sisters, then its leftmost sister will become the right daughter of x in the transformed tree. This transformation is applied recursively to all nodes in the tree. For example, the tree in Figure 2 is transformed into the binary tree in Figure 3.

Note how the disconnected punctuation node at the lower right corner in Figure 2 becomes the right daughter of the SIMPX node in Figure 3. Note also that we have both category and grammatical function as node labels for those nodes that have a grammatical function.

Such a binary tree is now described by a several formulae. The first formula, called *Carcase*, collects the addresses of all nodes in the tree to describe the tree structure without any labels. For our example tree, the formula would be

$$\text{Carcase} = \{root, root.0, root.00, root.000, root.0000, root.01, root.001, root.0010, root.00100, root.001001, root.0010010, root.0010011\}.$$

A syntactic category or grammatical function is coded as the set of nodes in the tree that are labelled

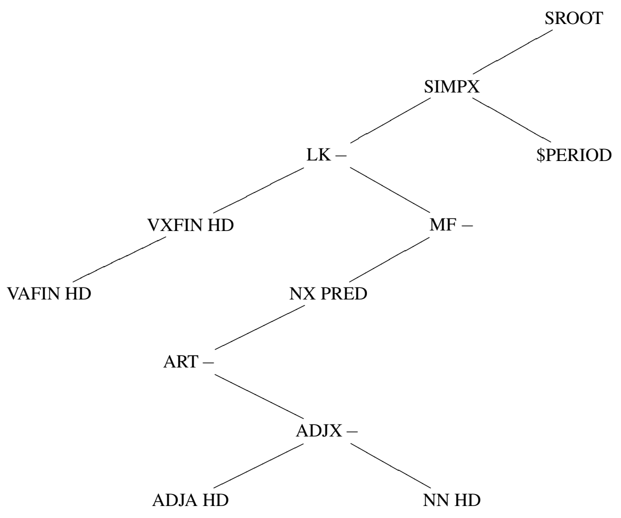


Figure 3: Binary recoding of the tree in Figure 2.

with this category or function. This is the way to circumvent the problem that we cannot extend the language of MONA. Here are some formulae for some labels of the example tree.
LK $= \{root.00\}$, ART $= \{root.00100\}$, HD $= \{root.000, root.0000, root.0010010, root.0010011\}$.

For all category or function labels that are not present at a particular tree, but part of the label set of the treebank, we state that the corresponding sets are empty. For example, the description of the example tree contains the formula `empty`(VC).

We implemented a program that actually performs this translation step. The input is a fraction of the TüBa-D/S treebank in NEGRA export format (Brants, 1997). The output is a file for each tree containing the MONA formulae coding this tree. In this way, we get a set of MONA formulae describing each tree.

4.2 Compiling Tree Descriptions into Automata

As mentioned above, the next step consists in compiling each tree description into an equivalent automaton. This is the first part of the evaluation. We tested whether MONA can actually perform this compilation. Astonishingly, the answer is not as simple as one might expect. It turns out that the computing power required to perform the compilation is quite high. To start, we chose a very small subset of the TüBa-D/S, just 1000 trees. Some of these trees contain more than 100 nodes, one more than 200 nodes. Processing descriptions of these large trees actually requires a lot of computing power.

It seems it is *not* possible to perform this compilation step on a desktop machine. We used an AMD 2200 machine with 2GB Ram for a try, but aborted the compilation of the 1000 trees after 15 hours. At that time, only 230 trees had been compiled.

To actually get through the compilation of the treebank we transfered the task to a cluster computer. On this cluster we used 4 nodes each equipped with two AMD Opteron 146 (2GHz, 4GB Ram) in parallel. Parallelisation is simple since each tree description can be compiled independently of all the others. The parallelisation was done by hand. Using this equipment we could compile 999 trees in about 4 hours. These 4 hours are the time needed to complete the whole task, not pure processing time. The tree containing more than 200 nodes could still not be compiled. Its compilation terminated unsuccessfully after 6 hours. We decided to drop this tree from the sample.

It is obvious that this is a major obstacle for using MONA. It is difficult to believe that many linguists will have access to a cluster computers and sufficient knowledge to use it. And we expect on the base of our experiences that a compilation on an ordinary desktop machine can take several days, provided the machine is equipped with large amounts of memory. Otherwise it will fail. One still has to consider that 1000 trees are not much. The TIGER corpus and the TüBa-D/S have each about 40.000 trees. Thus one may argue that this fact alone makes MONA unsuitable for use by linguists. But the compilation step has to be performed only once. The files containing the resulting automata are machine independent. Hence a corpus provider could at least in theory provide his corpus as a collection of MONA automata. This labour would be worth trying, if the resulting automata could be used for efficient querying.

5 Querying the Treebank

In order to query the treebank we designed a query language that has MSO as its core but contains features desirable for treebank querying. Naturally the language is designed to query the original trees, not their codings. It is therefore necessary to translate a query into an equivalent MONA formula that respects the translation of the trees.

5.1 The Query Language

The query language is defined as follows. The language has a LISP-like syntax. First-order variables ($x, y, \ldots$) range over nodes, set variables ($X, Y, \ldots$) range over sets of nodes. The atomic formulae are

- `(cat x NX)` – Node x is of category `NX`,
- `(fct x HD)` – Node x is of grammatical function `HD`,
- `(> x y)` – Node x is the mother of node y,
- `(>+ x y)` – Node x properly dominates y,
- `(. x y)` – Node x is immediately to the left of y,
- `(.. x y)` – Node x is to the left of y,
- `(= x y)` – Node x and y are identical,
- `(= X Y)` – Node sets X and Y are identical,
- `(in x X)` – Node x is a member of set X.

Complex formulae are constructed by boolean connectives and quantification. Let x be a node variable, X a set variable, and ϕ and ψ formulae. Then we have

- `(!` ϕ`)` – Negation of ϕ,
- `(&` ϕ ψ`)` – Conjunction of ϕ and ψ,
- `(|` ϕ ψ`)` – Disjunction of ϕ and ψ,
- `(->` ϕ ψ`)` – Implication of ϕ and ψ,
- `(E x` ϕ`)` – Existential quantification of x in ϕ,
- `(A x` ϕ`)` – Universal quantification of x in ϕ,
- `(E2 X` ϕ`)` – Existential quantification of set variable X in ϕ,
- `(A2 X` ϕ`)` – Universal quantification of set variable X in ϕ.

5.2 Translating the Query Language

The next step consists of translating queries in this language into MONA formulae. As is simple to see, the translation of the complex formulae is straight forward, because they are essentially the same in both languages. The more demanding task is connected with the translation of formulae on category and function labels and the tree structure, i.e., dominance and precedence.

As described above, categories and functions are coded as sets. Hence a query for a category or function is translated into a formula expressing set membership in the relevant set. For example, the query `(cat x SIMPX)` is translated into `(x in SIMPX)`.

The translations of dominance and precedence are the most complicated ones, because we transformed the treebank trees into binary trees. Now we have to reconstruct the original tree structures out of these binary trees. In the first step we have to define dominance on coded binary trees. The MONA language contains formulae for the left and right daughter of a node, but there is no formula for dominance, the transitive closure of the daughter relation. That we can define dominance at all is a consequence of the expressive power of MSO. As was shown by Courcelle (1990), the transitive closure of any MSO-definable binary relation is also MSO-definable. Let R be an MSO-definable binary relation. Then

$$\forall X \ (\forall z, w(z \in X \wedge R(z,w) \rightarrow w \in X) \wedge \\ \forall z(R(x,z) \rightarrow z \in X)) \rightarrow y \in X$$

is a formula with free variables x and y that defines the transitive closure of R. If we now take $R(x,y)$ in the above formula to be $(x.0 = y \mid x.1 = y)$ we define dominance (*dom*). In a similar fashion we can define that y is on the rightmost branch of x ($rb(x,y)$) by taking $R(x,y)$ to be $(x.1 = y)$.

Now for immediate dominance, if node x is the mother of y in the original tree, we have to distinguish to cases. In the simpler case, y is the leftmost daughter of x, so after transformation, y is the left daughter of x. Or y is not the leftmost daughter of x, in that case it is a sister of the leftmost daughter z of x. All sisters of z are found on the rightmost branch of z in the transformed trees. Hence `(> x y)` is translated into $(x.0 = y \mid \texttt{ex1}\ z : x.0 = z \,\&\, rb(z,y))$.

Proper dominance is treated similarly. If we iterate the above argument that the daughters of a node x in the original tree become the left daughter z of x and the rightmost successors of z, we can see that z and all the nodes dominated by z in the translated tree are actually all the nodes dominated by x in the original tree. Hence `(>+ x y)` is translated into $(x.0 = y \mid \texttt{ex1}\ z : x.0 = z \,\&\, dom(z,y))$.

For precedence, consider a node x in a coded binary tree. By definition the left daughter of x and all her successors are nodes that preceed the right daughter of x and her successors in the original tree. Thus `(.. x y)` is translated into

$$(x.1 = y \mid (\ \texttt{ex1}\ z,w,v : z.0 = w \,\&\, z.1 = v \,\& \\ (w = x \mid dom(w,x)) \,\& \\ (v = y \mid dom(v,y)))).$$

Immediate precedence can be expressed using precedence. Node x immediately precedes y if x precedes y there is no node z that is preceeded by x and precedes y.

There is a small issue in the translation of quantified formulae. In the translation of a first-order quantification (existential or universal) of a variable x we have to make sure that x actually ranges over the nodes in a particular tree. Otherwise MONA may construct an automaton that contains the coded tree as a substructure, but is more general. In such a case we could no longer be certain that a solution found by MONA actually represents a proper match of the original query on the original tree. To solve this problem, we add (x `in Carcase`) to the translation of `(E x` ϕ`)` or `(A x` ϕ`)`. E.g., `(E x` ϕ`)` translates to (`ex1` x : x `in Carcase` & ϕ') where ϕ' is the translation of ϕ. The same holds – mutatis mutandis – for set variable quantification.

5.3 Performing a Query

We implemented a small program that performs the above described translation of queries. It actually does a little bit more. It adds the defining formulae for *dom* and *rb*. Furthermore, as mentioned above, MONA allows to include a precompiled automaton into a set of MONA formulae via a special import declaration. Such an import declaration is used to include the automata representing the (coded) trees from the treebank. Thus the set of MONA formulae to evaluate a query consist of the translation of the query, the formulae for *dom* and *rb*, and an import declaration for one tree from the treebank. This set of MONA formulae can now be fed into MONA to try to compile it into an automaton. If the compilation is successful, there exists an automaton that at the same time represents the translation of the query and the translation of the given tree. Hence the tree is a match for the query. If there is no automaton, the tree is no match for the query. To perform the query on the whole treebank there is a loop that stepwise imports every tree and calls MONA to check if an automaton can be compiled. The result is the set of tree IDs that identify the trees that match the query.

We tested this method on our small treebank of 999 trees from TüBa-D/S. Unfortunately it turned out that the reloading of large precompiled automata (representing large trees) also requires enormous computational resources. We experimented with a very simple query: $\exists x\, \mathrm{NX}(x)$ (or `(E x (cat x NX)))`. On our desktop machine (AMD 2200, 2GB Ram), it took 6 hours and 9 minutes to process this query. If we pose the same query on the whole treebank TüBa-D/S (with about 38.000 trees) using established query tools like TIGERSearch or fsq, processing time is about 5 seconds. Hence the method of using MONA is clearly *not appropriate* for desktop computers.

Even access to larger computing power does not solve the problem. We processed the same query on one processor (AMD Opteron 146, 2GHz, 4GB Ram) of the cluster computer mentioned above. There it took 1 minute and 30 seconds. About the same query answer time was required for a second, more complex query that asked for two different NX nodes and a third SIMPX node. These query answer times are still too long, because we queried only about one fortieth of the whole treebank. Since each tree is queried separately, we can expect a linear time increase in the query time in the number of trees. In other words, evaluating the query on the whole treebank would probably take about 1 hour. And that on a computer with such massive computing power. TIGERSearch and fsq are 720 times faster, and they run on desktop computers.

6 Conclusions

Despite the many reported successful applications of MONA in other areas, we have to state that MONA is clearly *not* a choice for querying linguistic treebanks. Firstly, we cannot use MONA to query for tokens (or words). Secondly, the compilation of a treebank into a set of automata is extremely difficult and resources consuming, if not impossible. And finally, practical query answer times are way too long. Apparently, reloading precompiled automata representing large trees takes too much time, because the automata representing these large trees are themselves huge.

We note that this is unfortunately not the first negative experience of trying to apply MONA to computational linguistics tasks. Morawietz and Cornell (1999), who try to use MONA to compile logical formalisations of GB-theory, also report that automata get too large to work with.

The general problem behind these two unsuccessful applications of MONA to problems in computational linguistics seems to be that MONA does not allow users to define their own signatures. Hence linguistic labels have to be coded in an indirect fashion. Though this coding works in theory, the resulting automata can become huge. The reason for this explosion in automata size, though, remains mysterious.

The negative experience we made with MONA does on the other hand not mean that the whole enterprise of using tree automata for querying treebanks is deemed to fail. It seems that it is rather this particular deficit of MONA of providing no direct way to cope with labelled trees that causes the negative result. It could therefore well be worth trying to implement tree automata for labelled trees and use these for treebank querying.

References

Anne Abeillé and Lionel Clément. 1999. A tagged reference corpus for French. In *Proceedings of EACL-LINC*.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In Kiril Simov, editor, *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Thorsten Brants. 1997. The NEGRA export format. CLAUS Report 98, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany.

Bruno Courcelle. 1990. Graph rewriting: An algebraic and logic approach. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 5, pages 193–242. Elsevier.

Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol 3: Beyond Words*, pages 1–68. Springer-Verlag.

Erhard Hinrichs, Julia Bartels, Yasuhiro Kawata, Valia Kordoni, and Heike Telljohann. 2000. The VERBMOBIL treebanks. In *Proceedings of KONVENS 2000*.

Tilman Höhle. 1985. Der Begriff 'Mittelfeld'. Anmerkungen über die Theorie der topologischen Felder. In A. Schöne, editor, *Kontroversen, alte und neue. Akten des 7. Internationalen Germanistenkongresses*, pages 329–340.

Stephan Kepser. 2003. Finite Structure Query: A tool for querying syntactically annotated corpora. In Ann Copestake and Jan Hajič, editors, *Proceedings EACL 2003*, pages 179–186.

Stephan Kepser. 2004. Querying linguistic treebanks with monadic second-order logic in linear time. *Journal of Logic, Language, and Information*, 13:457–470.

Nils Klarlund. 1998. Mona & Fido: The logic-automaton connection in practice. In *Computer Science Logic, CSL '97*, LNCS 1414, pages 311–326. Springer.

Esther König and Wolfgang Lezius. 2000. A description language for syntactically annotated corpora. In *Proceedings of the COLING Conference*, pages 1056–1060.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Frank Morawietz and Tom Cornell. 1999. The MSO logic-automaton connection in linguistics. In Alain Lecomte, François Lamarche, and Guy Perrier, editors, *Logical Aspects of Computational Linguistics*, LNCS 1582, pages 112–131. Springer.

Beth Randall. 2000. CorpusSearch user's manual. Technical report, University of Pennsylvania. `http://www.ling.upenn.edu/mideng/ppcme2dir/`.

Douglas Rohde. 2001. Tgrep2. Technical report, Carnegie Mellon University.

Anne Schiller, Simone Teufel, and Christine Thielen. 1995. Guidelines für das Tagging deutscher Textcorpora mit STTS. Manuscript, Universities of Stuttgart and Tübingen.

Rosmary Stegmann, Heike Telljohann, and Erhard Hinrichs. 2000. Stylebook for the German treebank in VERBMOBIL. Technical Report 239, SfS, University of Tübingen.

Sean Wallis and Gerald Nelson. 2000. Exploiting fuzzy tree fragment queries in the investigation of parsed corpora. *Literary and Linguistic Computing*, 15(3):339–361.

KnowItNow: Fast, Scalable Information Extraction from the Web

Michael J. Cafarella, Doug Downey, Stephen Soderland, Oren Etzioni
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
{mjc,ddowney,soderlan,etzioni}@cs.washington.edu

Abstract

Numerous NLP applications rely on search-engine queries, both to extract information from and to compute statistics over the Web corpus. But search engines often limit the number of available queries. As a result, query-intensive NLP applications such as Information Extraction (IE) distribute their query load over several days, making IE a slow, offline process.

This paper introduces a novel architecture for IE that obviates queries to commercial search engines. The architecture is embodied in a system called KNOWITNOW that performs high-precision IE in minutes instead of days. We compare KNOWITNOW experimentally with the previously-published KNOWITALL system, and quantify the tradeoff between recall and speed. KNOWITNOW's extraction rate is two to three orders of magnitude higher than KNOWITALL's.

1 Background and Motivation

Numerous modern NLP applications use the Web as their corpus and rely on queries to commercial search engines to support their computation (Turney, 2001; Etzioni et al., 2005; Brill et al., 2001). Search engines are extremely helpful for several linguistic tasks, such as computing usage statistics or finding a subset of web documents to analyze in depth; however, these engines were not designed as building blocks for NLP applications. As a result, the applications are forced to issue literally millions of queries to search engines, which limits the speed, scope, and scalability of the applications. Further, the applications must often then fetch some web documents, which at scale can be very time-consuming.

In response to heavy programmatic search engine use, Google has created the "Google API" to shunt programmatic queries away from Google.com and has placed hard quotas on the number of daily queries a program can issue to the API. Other search engines have also introduced mechanisms to limit programmatic queries, forcing applications to introduce "courtesy waits" between queries and to limit the number of queries they issue.

To understand these efficiency problems in more detail, consider the KNOWITALL information extraction system (Etzioni et al., 2005). KNOWITALL has a generate-and-test architecture that extracts information in two stages. First, KNOWITALL utilizes a small set of domain-independent extraction patterns to *generate* candidate facts (*cf.* (Hearst, 1992)). For example, the generic pattern "NP1 such as NPList2" indicates that the head of each simple noun phrase (NP) in NPList2 is a member of the class named in NP1. By instantiating the pattern for class `City`, KNOWITALL extracts three candidate cities from the sentence: "We provide tours to cities such as Paris, London, and Berlin." Note that it must also fetch each document that contains a potential candidate.

Next, extending the PMI-IR algorithm (Turney, 2001), KNOWITALL automatically *tests* the plausibility of the candidate facts it extracts using *pointwise mutual information* (PMI) statistics computed from search-engine hit counts. For example, to assess the likelihood that "Yakima" is a city, KNOWITALL will compute the PMI between Yakima and a set of k *discriminator phrases* that tend to have high mutual information with city names (*e.g.*, the simple phrase "city"). Thus, KNOWITALL requires at least k search-engine queries for every candidate extraction it assesses.

Due to KNOWITALL's dependence on search-engine queries, large-scale experiments utilizing KNOWITALL *take days and even weeks to complete*, which makes research using KNOWITALL slow and cumbersome. Private access to Google-scale infrastructure would provide sufficient access to search queries, but at prohibitive cost, and the problem of fetching documents (even if from a cached copy) would remain (as we discuss in Section 2.1). Is there a feasible alternative Web-based IE system? If so, what size Web index and how many machines are required to achieve reasonable levels of precision/recall? What would the architecture of this IE system look like, and how fast would it run?

To address these questions, this paper introduces a novel architecture for web information extraction. It consists of two components that supplant the generate-and-test mechanisms in KNOWITALL. To generate extractions rapidly we utilize our own specialized search engine, called the Bindings Engine (or BE), which efficiently returns bindings in response to variabilized queries. For example, in response to the query *"Cities such as ProperNoun(Head(⟨NounPhrase⟩))"*, BE will return a list of proper nouns likely to be city names. To assess these extractions, we use URNS, a combinatorial model, which estimates the probability that each extraction is correct without using any additional search engine queries.[1] For further efficiency, we introduce an approximation to URNS, based on frequency of extractions' occurrence in the output of BE, and show that it achieves comparable precision/recall to URNS.

Our contributions are as follows:

1. We present a novel architecture for Information Extraction (IE), embodied in the KNOWITNOW system, which does not depend on Web search-engine queries.

2. We demonstrate experimentally that KNOWITNOW is the first system able to extract tens of thousands of facts from the Web in minutes instead of days.

3. We show that KNOWITNOW's extraction rate is two to three orders of magnitude greater than KNOWITALL's, but this increased efficiency comes at the cost of reduced recall. We quantify this tradeoff for KNOWITNOW's 60,000,000 page index and extrapolate how the tradeoff would change with larger indices.

Our recent work has described the BE search engine in detail (Cafarella and Etzioni, 2005), and also analyzed the URNS model's ability to compute accurate probability estimates for extractions (Downey et al., 2005). However, this is the first paper to investigate the composition of these components to create a fast IE system, and to compare it experimentally to KNOWITALL in terms of time, recall, precision, and extraction rate. The frequency-based approximation to URNS and the demonstration of its success are also new.

The remainder of the paper is organized as follows. Section 2 provides an overview of BE's design. Section 3 describes the URNS model and introduces an efficient approximation to URNS that achieves similar precision/recall. Section 4 presents experimental results. We conclude with related and future work in Sections 5 and 6.

2 The Bindings Engine

This section explains how relying on standard search engines leads to a bottleneck for NLP applications, and provides a brief overview of the Bindings Engine (BE)—our solution to this problem. A comprehensive description of BE appears in (Cafarella and Etzioni, 2005).

Standard search engines are computationally expensive for IE and other NLP tasks. IE systems issue multiple queries, downloading all pages that potentially match an extraction rule, and performing expensive processing on each page. For example, such systems operate roughly as follows on the query ("cities such as ⟨*NounPhrase*⟩"):

1. Perform a traditional search engine query to find all URLs containing the non-variable terms (*e.g.*, "cities such as")

2. For each such URL:
   (a) obtain the document contents,
   (b) find the searched-for terms ("cities such as") in the document text,
   (c) run the noun phrase recognizer to determine whether text following "cities such as" satisfies the linguistic type requirement,
   (d) and if so, return the string

We can divide the algorithm into two stages: obtaining the list of URLs from a search engine, and then processing them to find the ⟨*NounPhrase*⟩ bindings. Each stage poses its own scalability and speed challenges. The first stage makes a query to a commercial search engine; while the number of available queries may be limited, a single one executes relatively quickly. The second stage fetches a large number of documents, each fetch likely resulting in a random disk seek; this stage executes slowly. Naturally, this disk access is slow regardless of whether it happens on a locally-cached copy or on a remote document server. The observation that the second stage is slow, even if it is executed locally, is important because it shows that merely operating a "private" search engine does not solve the problem (see Section 2.1).

The Bindings Engine supports queries containing *typed variables* (such as *NounPhrase*) and

[1]In contrast, PMI-IR, which is built into KNOWITALL, requires multiple search engine queries to assess each potential extraction.

string-processing functions (such as "head(X)" or "ProperNoun(X)") as well as standard query terms. BE processes a variable by returning every possible string in the corpus that has a matching type, and that can be substituted for the variable and still satisfy the user's query. If there are multiple variables in a query, then all of them must simultaneously have valid substitutions. (So, for example, the query "<*NounPhrase*> is located in <*NounPhrase*>" only returns strings when noun phrases are found on both sides of "is located in".) We call a string that meets these requirements a *binding* for the variable in question. These queries, and the bindings they elicit, can usefully serve as part of an information extraction system or other common NLP tasks (such as gathering usage statistics). Figure 1 illustrates some of the queries that BE can handle.

president Bush <*Verb*>
cities such as *ProperNoun(Head(<NounPhrase>))*
<*NounPhrase*> is the CEO of <*NounPhrase*>

Figure 1: **Examples of queries that can be handled by** BE**. Queries that include typed variables and string-processing functions allow NLP tasks to be done efficiently without downloading the original document during query processing.**

BE's novel *neighborhood index* enables it to process these queries with $O(k)$ random disk seeks and $O(k)$ serial disk reads, where k is the number of non-variable terms in its query. As a result, BE can yield orders of magnitude speedup as shown in the asymptotic analysis later in this section. The neighborhood index is an augmented inverted index structure. For each term in the corpus, the index keeps a list of documents in which the term appears and a list of positions where the term occurs, just as in a standard inverted index (Baeza-Yates and Ribeiro-Neto, 1999). In addition, the neighborhood index keeps a list of left-hand and right-hand *neighbors* at each position. These are adjacent text strings that satisfy a recognizer for one of the target types, such as *NounPhrase*.

As with a standard inverted index, a term's list is processed from start to finish, and can be kept on disk as a contiguous piece. The relevant string for a variable binding is included directly in the index, so there is no need to fetch the source document (thus causing a disk seek). Expensive processing such as part-of-speech tagging or shallow syntactic parsing is performed only once, while building the index, and is not needed at query time. It is important to note that simply preprocessing the corpus and placing the results in a database would not avoid disk seeks, as we would still have to explicitly fetch these results. The run-time efficiency of the neighborhood index comes from integrating the results of corpus processing with the inverted index (which determines which of those results are relevant).

	Query Time	Index Space
BE	$O(k)$	$O(N)$
Standard engine	$O(k+B)$	$O(N)$

Table 1: BE **yields considerable savings in query time over a standard search engine. k is the number of concrete terms in the query, B is the number of variable bindings found in the corpus, and N is the number of documents in the corpus. N and B are typically extremely large, while k is small.**

The neighborhood index avoids the need to return to the original corpus, but it can consume a large amount of disk space, as parts of the corpus text are folded into the index several times. To conserve space, we perform simple dictionary-lookup compression of strings in the index. The storage penalty will, of course, depend on the exact number of different types added to the index. In our experiments, we created a useful IE system with a small number of types (including *NounPhrase*) and found that the neighborhood index increased disk space only four times that of a standard inverted index.

Asymptotic Analysis:

In our asymptotic analysis of BE's behavior, we count query time as a function of the number of random disk seeks, since these seeks dominate all other processing tasks. Index space is simply the number of bytes needed to store the index (not including the corpus itself).

Table 1 shows that BE requires only $O(k)$ random disk seeks to process queries with an arbitrary number of variables whereas a standard engine takes $O(k + B)$, where k is the number of concrete query terms, and B is the number of bindings found in a corpus of N documents. Thus, BE's performance is the same as that of a standard search engine for queries containing only concrete terms. For variabilized queries, N may be in the billions and B will tend to grow with N. In our experiments, eliminating the B term from our query processing time has resulted in speedups of two to three orders of magnitude over a standard search engine. The speedup is at the price of a small constant multiplier to index size.

2.1 Discussion

While BE has some attractive properties for NLP computations, is it necessary? Could fast, large-scale information extraction be achieved merely by operating a "private" search engine?

The release of open-source search engines such as Nutch[2], coupled with the dropping price of CPUs and

[2]http://lucene.apache.org/nutch/

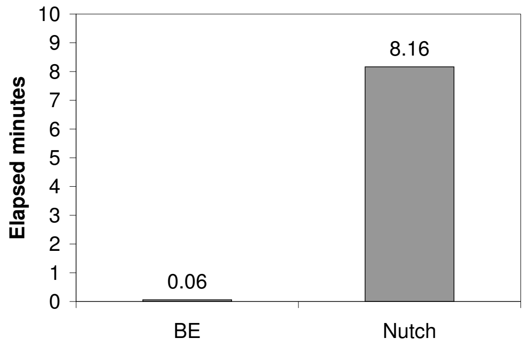


Figure 2: **Average time to return the relevant bindings in response to a set of queries was 0.06 CPU minutes for BE, compared to 8.16 CPU minutes for the comparable processing on Nutch. This is a 134-fold speed up. The CPU resources, network, and index size were the same for both systems.**

disks, makes it feasible for NLP researchers to operate their own large-scale search engines. For example, Turney operates a search engine with a terabyte-sized index of Web pages, running on a local eight-machine Beowulf cluster (Turney, 2004). Private search engines have two advantages. First, there is no query quota or need for "courtesy waits" between queries. Second, since the engine is local, network latency is minimal.

However, to support IE, we must also execute the second stage of the algorithm (see the beginning of this section). In this stage, each document that matches a query has to be retrieved from an arbitrary location on a disk.[3] Thus, the number of random disk seeks scales linearly with the number of documents retrieved. Moreover, many NLP applications require the extraction of strings matching particular syntactic or semantic types from each page. The lack of linguistic data in the search engine's index means that many pages are fetched only to be discarded as irrelevant.

To quantify the speedup due to BE, we compared it to a standard search index built on the open-source Nutch engine. All of our Nutch and BE experiments were carried out on the same corpus of 60 million Web pages and were run on a cluster of 23 dual-Xeon machines, each with two local 140 Gb disks and 4 Gb of RAM. We set all configuration values to be exactly the same for both Nutch and BE. BE gave a 134-fold speed up on average query processing time when compared to the same queries with the Nutch index, as shown in Figure 2.

[3]Moving the disk head to an arbitrary location on the disk is a mechanical operation that takes about 5 milliseconds on average.

3 The URNS Model

To realize the speedup from BE, KNOWITNOW must also avoid issuing search engine queries to validate the correctness of each extraction, as required by PMI computation. We have developed a probabilistic model obviating search-engine queries for assessment. The intuition behind this model is that correct instances of a class or relation are likely to be extracted repeatedly, while random errors by an IE system tend to have low frequency for each distinct incorrect extraction.

Our probabilistic model, which we call URNS, takes the form of a classic "balls-and-urns" model from combinatorics. We think of IE abstractly as a generative process that maps text to extractions. Each extraction is modeled as a labeled ball in an urn. A *label* represents either an instance of the target class or relation, or represents an error. The information extraction process is modeled as repeated draws from the urn, with replacement.

Formally, the parameters that characterize an urn are:

- C – the set of unique target labels; $|C|$ is the number of unique target labels in the urn.
- E – the set of unique error labels; $|E|$ is the number of unique error labels in the urn.
- $num(b)$ – the function giving the number of balls labeled by b where $b \in C \cup E$. $num(B)$ is the multi-set giving the number of balls for each label $b \in B$.

The goal of an IE system is to discern which of the labels it extracts are in fact elements of C, based on repeated draws from the urn. Thus, the central question we are investigating is: *given that a particular label x was extracted k times in a set of n draws from the urn, what is the probability that $x \in C$?* We can express the probability that an element extracted k of n times is of the target relation as follows.

$$P(x \in C | x \, \text{appears} \, k \, \text{times in} \, n \, \text{draws}) = \frac{\sum_{r \in num(C)} (\frac{r}{s})^k (1 - \frac{r}{s})^{n-k}}{\sum_{r' \in num(C \cup E)} (\frac{r'}{s})^k (1 - \frac{r'}{s})^{n-k}} \quad (1)$$

where s is the total number of balls in the urn, and the sum is taken over possible repetition rates r.

A few numerical examples illustrate the behavior of this equation. Let $|C| = |E| = 2,000$ and assume for simplicity that all labels are repeated on the same number of balls ($num(c_i) = R_C$ for all $c_i \in C$, and $num(e_i) = R_E$ for all $e_i \in E$). Assume that the extraction rules have precision $p = 0.9$, which means that $R_C = 9 \times R_E$ — target balls are nine times as common in the urn as error balls. Now, for $k = 3$ and $n = 10,000$ we have $P(x \in C) = 93.0\%$. Thus, we see that a small number of repetitions can yield high confidence in an extraction. However, when the sample size increases so that

$n = 20,000$, and the other parameters are unchanged, then $P(x \in C)$ drops to 19.6%. On the other hand, if C balls repeat much more frequently than E balls, say $R_C = 90 \times R_E$ (with $|E|$ set to 20,000, so that p remains unchanged), then $P(x \in C)$ rises to 99.9%.

The above examples enable us to illustrate the advantages of URNS over the noisy-or model used in previous work. The noisy-or model assumes that each extraction is an independent assertion that the extracted label is "true," an assertion that is correct a fraction p of the time. The noisy-or model assigns the following probability to extractions:

$$P_{noisy-or}(x \in C | x \text{ appears } k \text{ times}) = 1 - (1-p)^k$$

Therefore, the noisy-or model will assign the same probability — 99.9% — in *all three* of the above examples, although this is only correct in the case for which $n = 10,000$ and $R_C = 90 \times R_E$. As the other two examples show, for different sample sizes or repetition rates, the noisy-or model can be highly inaccurate. This is not surprising given that the noisy-or model ignores the sample size and the repetition rates.

URNS uses an EM algorithm to estimate its parameters, and currently the algorithm takes roughly three minutes to terminate.[4] Fortunately, we determined experimentally that we can approximate URNS's precision and recall using a far simpler frequency-based assessment method. This is true because good precision and recall merely require an appropriate *ordering* of the extractions for each relation, and not accurate probabilities for each extraction. For unary relations, we use the simple approximation that items extracted more often are more likely to be true, and order the extractions from most to least extracted. For binary relations like CapitalOf(X,y), in which we extract several different candidate capitals y for each known country X, we use a smoothed frequency estimate to order the extractions. Let $freq(\mathtt{R(X,y)})$ denote the number of times that the binary relation $\mathtt{R(X,y)}$ is extracted; we define:

$$smoothed\_freq(\mathtt{R(X,y)}) = \frac{freq(\mathtt{R(X,y)})}{\max_{y'} freq(\mathtt{R(X,y')}) + 1}$$

We found that sorting by smoothed frequency (in descending order) performed better than simply sorting by $freq$ for relations $\mathtt{R(X,y)}$ in which different known X values may have widely varying Web presence.

Unlike URNS, our frequency-based assessment does not yield accurate probabilities to associate with each extraction, but for the purpose of returning a ranked list of high-quality extractions it is comparable to URNS (see Figures 3 through 6), and it has the advantage of being much faster. Thus, in the experiments reported on below, we use frequency-based assessment as part of KNOWITNOW.

[4]This code has not been optimized at all. We believe that we can easily reduce its running time to less than a minute on average, and perhaps substantially more.

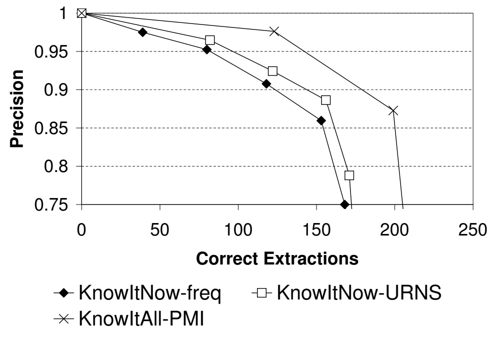


Figure 3: Country: **KNOWITALL maintains somewhat higher precision than KNOWITNOW throughout the recall-precision curve.**

4 Experimental Results

This section contrasts the performance of KNOWITNOW and KNOWITALL experimentally. Before considering the experiments in detail, we note that a key advantage of KNOWITNOW is that it does not make *any* queries to Web search engines. As a result, KNOWITNOW's scale is not limited by a query quota, though it is limited by the size of its index.

We report on the following metrics:

- **Recall:** how many distinct extractions does each system return at high precision?[5]
- **Time:** how long did each system take to produce and rank its extractions?
- **Extraction Rate:** how many distinct high-quality extractions does the system return per minute? The extraction rate is simply recall divided by time.

We contrast KNOWITALL and KNOWITNOW's precision/recall curves in Figures 3 through 6. We compared KNOWITNOW with KNOWITALL on four relations: Corp, Country, CeoOf(Corp,Ceo), and CapitalOf(Country,City). The unary relations were chosen to examine the difference between a relation with a small number of correct instances (Country) and one with a large number of extractions (Corp). The binary relations were chosen to cover both functional relations (CapitalOf) and set-valued relations (CeoOf— we treat former CEOs as correct instances of the relation).

[5]Since we cannot compute "true recall" for most relations on the Web, the paper uses the term "recall" to refer to the size of the set of facts extracted.

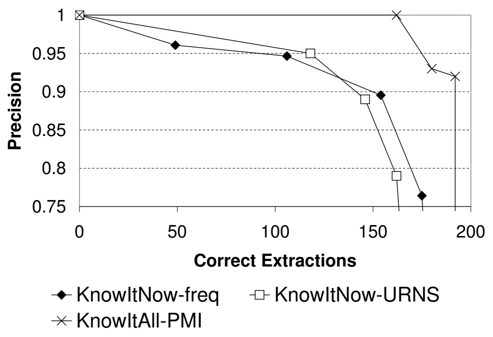


Figure 4: `CapitalOf`: KNOWITNOW **does nearly as well as** KNOWITALL**, but has more difficulty than** KNOWITALL **with sparse data for capitals of more obscure countries.**

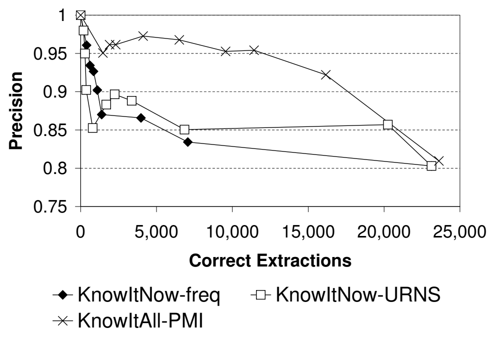


Figure 5: `Corp`: KNOWITALL**'s PMI assessment maintains high precision.** KNOWITNOW **has low recall up to precision 0.85, then catches up with** KNOWITALL**.**

For the two unary relations, both systems created extraction rules from eight generic patterns. These are hyponym patterns like "NP1 {,} such as NPList2" or "NP2 {,} and other NP1", which extract members of NPList2 or NP2 as instances of NP1. For the binary relations, the systems instantiated rules from four generic patterns. These are patterns for a generic "of" relation. They are "NP1 , rel of NP2", "NP1 the rel of NP2", "rel of NP2 , NP1", and "NP2 rel NP1". When rel is instantiated for `CeoOf`, these patterns become "NP1 , CEO of NP2" and so forth.

Both KNOWITNOW and KNOWITALL merge extractions with slight variants in the name, such as those differing only in punctuation or whitespace, or in the presence or absence of a corporate designator. For binary extractions, CEOs with the same last name and same company were also merged. Both systems rely on the OpenNlp maximum-entropy part-of-speech tagger and chunker (Ratnaparkhi, 1996), but KNOWITALL applies them to pages downloaded from the Web based on the results of Google queries, whereas KNOWITNOW applies them once to crawled and indexed pages.[6] Overall, each of the above elements of KNOWITALL and KNOWITNOW are the same to allow for controlled experiments.

Whereas KNOWITNOW runs a small number of variabilized queries (one for each extraction pattern, for each relation), KNOWITALL requires a stopping criterion. Otherwise, KNOWITALL will continue to query Google and download URLs found in its result pages over many days and even weeks. We allowed a total of 6 days of search time for KNOWITALL, allocating more search for the relations that continued to be most productive. For `CeoOf` KNOWITNOW returned all pairs of `Corp,Ceo` in its corpus; KNOWITALL searched for CEOs of a random selection of 10% of the corporations it found, and we projected the total extractions and search effort for all corporations. For `CapitalOf`, both KNOWITNOW and KNOWITALL looked for capitals of a set of 195 countries.

Table 2 shows the number of queries, search time, distinct correct extractions at precision 0.8, and extraction rate for each relation. Search time for KNOWITNOW is measured in seconds and search time for KNOWITALL is measured in hours. The number of extractions per minute counts the distinct correct extractions. Since we limit KNOWITALL to one Google query per second, the time for KNOWITALL is proportional to the number of queries. KNOWITNOW's extraction rate is from 275 to 4,707 times that of KNOWITALL at this level of precision.

While the number of distinct correct extractions from KNOWITNOW at precision 0.8 is roughly comparable to that of 6 days search effort from KNOWITALL, the situation is different at precision 0.9. KNOWITALL's PMI assessor is able to maintain higher precision than KNOWITNOW's frequency-based assessor. The number of correct corporations for KNOWITNOW drops from 23,128 at precision 0.8 to 1,116 at precision 0.9. KNOWITALL is able to identify 17,620 correct corporations at precision 0.9. Even with the drop in recall, KNOWITNOW's extraction rate is still 305 times higher than KNOWITALL's. The reason for KNOWITNOW's difficulty at precision 0.9 is due to extraction errors that occur with high frequency, particularly generic references to companies ("the Seller is a corporation ...", "corporations such as Banks", etc.) and truncation of certain company names by the extraction rules. The more expensive PMI-based assessment was not fooled by these systematic extraction errors.

Figures 3 through 6 show the recall-precision curves for KNOWITNOW with URNS assessment, KNOWITNOW with the simpler frequency-based assessment, and

[6]Our time measurements for KNOWITALL are not affected by the tagging and chunking time because it is dominated by time required to query Google, waiting a second between queries.

	Google Queries		Time		Extractions		Extractions per minute		
	NOW	ALL	NOW (sec)	ALL (hrs)	NOW	ALL	NOW	ALL	ratio
Corp	0 (16)	201,878	42	56.1	23,128	23,617	33,040	7.02	4,707
Country	0 (16)	35,480	42	9.9	161	203	230	0.34	672
CeoOf	0 (6)	263,646	51	73.2	2,402	5,823	2,836	1.33	2,132
CapitalOf	0 (6)	17,216	55	4.8	169	192	184	0.67	275

Table 2: **Comparison of KNOWITNOW with KNOWITALL for four relations, showing number of Google queries (local BE queries in parentheses), search time, correct extractions at precision 0.8, and extraction rate (the number of correct extractions at precision 0.8 per minute of search). Overall, KNOWITNOW took a total of slightly over 3 minutes as compared to a total of 6 days of search for KNOWITALL.**

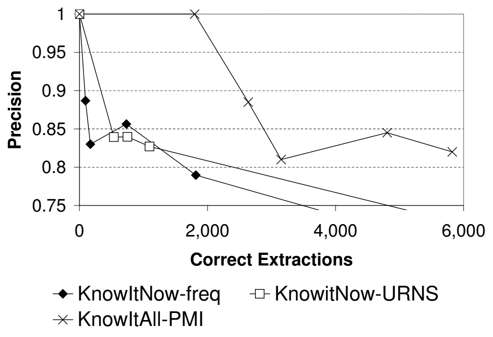


Figure 6: `CeoOf`: **KNOWITNOW has difficulty distinguishing low frequency correct extractions from noise. KNOWITALL is able to cope with the sparse data more effectively.**

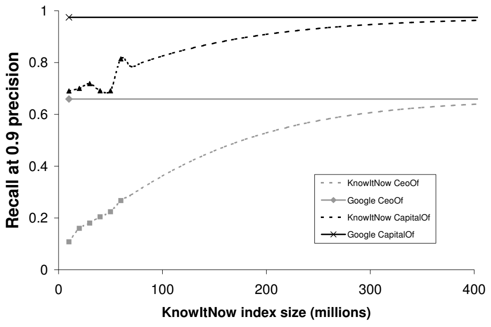


Figure 7: **Projections of recall (at precision 0.9) as a function of KNOWITNOW index size. At 400 million pages, KNOWITNOW's recall rapidly approaches the recall achieved by KNOWITALL using roughly 300,000 Google queries.**

KNOWITALL with PMI-based assessment. For each of the four relations, PMI is able to maintain a higher precision than either frequency-based or URNS assessment. URNS and frequency-based assessment give roughly the same levels of precision.

For the relations with a small number of correct instances, `Country` and `CapitalOf`, KNOWITNOW is able to identify 70-80% as many instances as KNOWITALL at precision 0.9. In contrast, `Corp` and `CeoOf` have a huge number of correct instances and a long tail of low frequency extractions that KNOWITNOW has difficulty distinguishing from noise. Over one fourth of the corporations found by KNOWITALL had Google hit counts less than 10,500, a sparseness problem that was exacerbated by KNOWITNOW's limited index size.

Figure 7 shows projected recall from larger KNOWITNOW indices, fitting a sigmoid curve to the recall from index size of 10M, 20M, up to 60M pages. The curve was fitted using logistic regression, and is restricted to asymptote at the level reported for Google-based KNOWITALL for each relation. We report recall at precision 0.9 for capitals of 195 countries and CEOs of a random selection of the top 5,000 corporations as ranked by PMI. Recall is defined as the percent of countries with a correct capital or the number of correct CEOs divided by the number of corporations.

The curve for `CeoOf` is rising steeply enough that a 400 million page KNOWITNOW index may approach the same level of recall yielded by KNOWITALL when it uses 300,000 Google queries. As shown in Table 2, KNOWITALL takes slightly more than three days to generate these results. KNOWITNOW would operate over a corpus 6.7 times its current one, but the number of required random disk seeks (and the asymptotic run time analysis) would remain the same. We thus expect that with a larger corpus we can construct a KNOWITNOW system that reproduces KNOWITALL levels of precision and recall while still executing in the order of a few minutes.

5 Related Work

There has been very little work published on how to make NLP computations such as PMI-IR and IE fast for large corpora. Indeed, extraction rate is not a metric typically used to evaluate IE systems, but we believe it is an important metric if IE is to scale.

Hobbs *et al.* point out the advantage of fast text processing for rapid system development (Hobbs et al., 1992). They could test each change to system parameters and domain-specific patterns on a large sample of documents, having moved from a system that took 36 hours to process 100 documents to FASTUS, which took only 11 minutes. This allowed them to develop one of the highest performing MUC-4 systems in only one month.

While there has been extensive work in the IR and Web communities on improvements to the standard inverted index scheme, there has been little work on efficient large-scale search to support natural language applications. One exception is Resnik's Linguist's Search Engine (Elkiss and Resnik, 2004), a tool for searching large corpora of parse trees. There is little published information about its indexing system, but the user manual suggests its corpus is a combination of indexed sentences and user-specific document collections driven by the user's AltaVista queries. In contrast, the BE system has a single index, constructed just once, that serves all queries. There is no published performance data available for Resnik's system.

6 Conclusions and Future Directions

In previous work, statistical NLP computation over large corpora has been a slow, offline process, as in KNOWITALL (Etzioni et al., 2005) and also in PMI-IR applications such as sentiment classification (Turney, 2002). Technology trends, and open source search engines such as Nutch, have made it feasible to create "private" search engines that index large collections of documents; but as shown in Figure 2, firing large numbers of queries at private search engines is still slow.

This paper described a novel and practical approach towards substantially speeding up IE. We described KNOWITNOW, which extracts thousands of facts in minutes instead of days. Furthermore, we sketched URNS, a probabilistic model that both obviates the need for search-engine queries and outputs more accurate probabilities than PMI-IR. Finally, we introduced a simple, efficient approximation to URNS, whose probability estimates are not as good, but which has comparable precision/recall to URNS, making it an appropriate assessor for KNOWITNOW.

The speed and massively improved extraction rate of KNOWITNOW come at the cost of reduced recall. We quantified this tradeoff in Table 2, and also argued that as KNOWITNOW's index size increases from 60 million to 400 million pages, KNOWITNOW would achieve in minutes the same precision/recall that takes KNOWITALL days to obtain. Of course, a hybrid approach is possible where KNOWITNOW has, say, a 100 million page index and, when necessary, augments its results with a limited number of queries to Google. Investigating the extraction-rate/recall tradeoff in such a hybrid system is a natural next step.

While our experiments have used the Web corpus, our approach transfers readily to other large corpora; experimentation with other corpora is another topic for future work. In conclusion, we believe that our techniques transform IE from a slow, offline process to an online one. They could open the door to a new class of interactive IE applications, of which KNOWITNOW is merely the first.

7 Acknowledgments

This research was supported in part by NSF grant IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-02-1-0324, and gifts from Google and the Turing Center.

References

R. Baeza-Yates and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley.

E. Brill, J. Lin, M. Banko, S. T. Dumais, and A. Y. Ng. 2001. Data-intensive question answering. In *Procs. of Text REtrieval Conference (TREC-10)*, pages 393–400.

M. Cafarella and O. Etzioni. 2005. A Search Engine for Natural Language Applications. In *Procs. of the 14th International World Wide Web Conference (WWW 2005)*.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *Procs. of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*.

E. Elkiss and P. Resnik, 2004. *The Linguist's Search Engine User's Guide*. University of Maryland.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France.

J.R. Hobbs, D. Appelt, M. Tyson, J. Bear, and D. Israel. 1992. Description of the FASTUS system used for MUC-4. In *Procs. of the Fourth Message Understanding Conference*, pages 268–275.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Procs. of the Empirical Methods in Natural Language Processing Conference*, Univ. of Pennsylvania.

P. D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Procs. of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, Freiburg, Germany.

P. D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Procs. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424.

P. D. Turney, 2004. *Waterloo MultiText System*. Institute for Information Technology, Nat'l Research Council of Canada.

A Cost-Benefit Analysis of Hybrid Phone-Manner Representations for ASR

Eric Fosler-Lussier
Department of Computer Science and Engineering
Department of Linguistics
Ohio State University
Columbus, OH 43210
fosler@cse.ohio-state.edu

C. Anton Rytting
Department of Linguistics
Ohio State University
Columbus, OH 43210
rytting@ling.ohio-state.edu

Abstract

In the past decade, several researchers have started reinvestigating the use of sub-phonetic models for lexical representations within automatic speech recognition systems. Lest history repeat itself, it may be instructive to mine the further past for models of lexical representations in the lexical access literature. In this work, we re-evaluate the model of Briscoe (1989), in which a hybrid strategy of lexical representation between phones and manner classes is promoted. While many of Briscoe's assumptions do not match up with current ASR processing models, we show that his conclusions are essentially correct, and that reconsidering this structure for ASR lexica is an appropriate avenue for future ASR research.

1 Introduction

Almost every state-of-the-art large vocabulary automatic speech recognition (ASR) system requires the sharing of sub-word units in order to achieve the desired vocabulary coverage. Traditionally, these sub-word units are determined by the phones or phonemes of a language (depending on desired detail of representation). However, phonetic (or phonemic) representation has its pitfalls (*cf*. (Ostendorf, 1999)). Among the problems cited in the literature are that (1) segments are often difficult for machines to recognize from the acoustic cues alone, because the acoustic cues to a particular phoneme are multi-faceted, and (2) the intended words and phrases are not always recoverable even from correctly recognized segments, because speakers themselves will also fail to articulate words with the dictionary-listed phonemes. The first of these problems refers to the *discriminability* of phonemes within an inventory; the second to the *reliability* of (actual) phone sequences mapping to the canonical phonemic representations of words. This is particularly true in conversational speech (such as that found in the Switchboard corpus), where pragmatic context and conversational conventions assist human comprehension (but not current ASR systems).

A common approach for handling pronunciation variation is to introduce alternative entries into the lexicon. However, phones that are perceived as non-canonical (for example, when an /eh/ is heard as an /ih/ by linguistic transcribers) often are closer in acoustic space to the Gaussian means of the canonical phones, rather than the perceived phones (Saraçlar et al., 2000). This insight suggests that acoustic models need to be cognizant of potential pronunciation changes. Thus the lexical and acoustic models should work hand in hand.

Another way to model this type of pronunciation variation is to find the commonalities that the canonical and perceived phone share in terms of a sub-phonetic representation. In the past decade, a significant community in acoustic-phonetic ASR research has been turning to distinctive features (Jakobson et al., 1952) for building ASR lexica. While an exhaustive description of these approaches is beyond the scope of this paper, estimates of phonological feature probabilities have been combined to obtain phone probabilities (Kirchhoff, 1998), or incorporated into "feature bundles" that allow representation of phonological processes (Finke et al., 1999).

More recent work has integrated phonological features into graphical models (Livescu et al., 2003) and landmark based systems (Juneja and Espy-Wilson, 2004). The common thread among this research is the notion that acoustic models should be sensitive to sub-phonetic information. With this trend in phonological representation research, it is time to re-examine some older hypotheses about lexical access and speech processing in order to gain some insight in this current featural renaissance.

Sub-phonetic ASR research is also driven by the fact that deviations from canonical pronunciation and from correct perception of phones is far from random; indeed, there have been a number of studies demonstrating that both of these variations have defined, modelable trends. Deviations from canonical pronunciation can be described by phonological rules, and errors in perception also tend to conform to phonological patterns. By and large, confusions occur (at least in humans) between phones with phonological features in common (e.g., (Miller and Nicely, 1955)). In particular, three features (voicing, manner, and place) have been postulated as relatively invariant (see e.g., (Stevens, 1981), quoted in (Church, 1987)). It follows from this phonetic detection based on the most reliable features may handle highly variable speech more robustly than systems which demand full identity over all the features for a given phone or phone sequence.

Consequently, a number of researchers have previously suggested using certain broad classes of segments, rather than full phonemic identification, for a first pass on recognition. For instance, Shipman and Zue (1982), working on large-vocabulary isolated word recognition, used both two-way consonant-vowel distinctions and a six-way distinction based on manner in order to divide their 20,000-word dictionary into “cohorts” or groups of words. They found that this partial specification of segments reduced the search space of word candidates significantly. Carlson et al. (1985) found similar results for English and four other languages.

2 A suggested compromise: a hybrid phone-manner representation

Briscoe (1989) extended this broad-class approach to address the problem of lexical access on connected speech. However, Briscoe argues against the use of broad, manner-based classes at all times. He argues that manner cues provide no particular advantage for stressed syllables, but that all cues are sufficiently reliable in stressed syllables to justify a full segmental analysis. Working with a 30,000-word lexicon, Briscoe shows that the manner-based broad classes for weak (reduced) syllables, together with full identification of strong (unreduced) syllables constrained the set of possible candidates satisfactorily. Unfortunately, he only provides results for one sentence from his corpus.

This approach proposes to adjust the granularity of recognition dynamically, depending on the stress level of the current syllable. The details of how this would be managed are left somewhat vague. As it stands, it would seem to depend crucially on first detecting the stress of each frame, so as to determine which alphabet of symbols to apply to incoming input. Alternatively, it could recognize the broad class as a first pass, and then refine this into a full phonemic analysis for stressed syllables in a second pass, at the cost of multiplying passes through the speech data. It is not possible in this system to recover from the miscategorization of stress.

One possible remedy is to bypass a hard decision on stress and run both a manner-based broad-class detector and a traditional phonemic system in parallel. These then may be combined according to the probability of lexical stress, such that those frames judged less likely to be stressed weight the broad-class analysis more heavily, and those judged more likely to be stressed weight the narrow phonemic analysis more heavily. Its advantage is that a full phonemic analysis is recoverable for each frame and phone, but those in weak syllables (and hence less likely to be accurate) weigh in less heavily.

Briscoe’s analysis is in terms of lexical access activations: taking a cue from the lexical access community, he assumes that any “partially activated” word (e.g., “boat” and “both” being active after processing “bo”) will contribute linearly to the processing time in ASR. However, most large-vocabulary ASR systems today use a tree-based lexicon where common phonetic prefixes of words are processed only once, thus invalidating this conjecture. Briscoe experimented with several triggers for starting a new word — at every phone, at the beginnings of syllables, at the beginnings of syllables with unreduced vowels, and at the beginnings of word boundaries.

Category	Example (Vietnamese)	Cohorts
# of Words		6247
Stress pattern only (lower bound)	0001	84
Identify phones in stressed syllables	0001:m_iy_z	4709
+CV pattern of unstressed syllables	0001:CV:VC:CV:m_iy_z	5609
+manner pattern of unstressed syllables	0001:FV:CS:NV:m_iy_z	6076
Phonetic prons. (upper bound)	v iy . eh t . n aa . m iy z	6152

Table 1: Cohorts for varied lexical representations

However, the latter three require oracle information as to where word or syllable boundaries can occur. A more appropriate measure commensurate with current ASR practice would be to only allow words to start where a previous word hypothesis ends.

In the remainder of the paper, we seek to validate (or invalidate) Briscoe's claim that a hybrid phonetic and feature model is appropriate for ASR processing. In the 15 years since Briscoe's paper, the ASR community has developed large phonetically transcribed corpora and more advanced computational tools (such as the AT&T Finite-State Toolkit (Mohri et al., 2001)) that we can apply to this problem.

3 Experiment 1: Effective Partitioning by Manner-based Broad Classes

Our first experiment explores various types of broad classes to determine the effects of these encodings on cohort size within a sample 6,000 word dictionary.[1] Here we use the lexical stress-marked dictionary provided with the TIMIT database (Garofolo et al., 1993), which was syllabified using the NIST Tsylb2 syllabifier (Fisher, 1996).

Rather than calculate cohort size directly, we calculate the number of cohorts into which our dictionary is partitioned, a measure which Carlson et al. (1985) showed to correlate well with expected cohort size. (Note that this is an inverse correlation.) This describes the static *discriminability* of the lexicon: systems that have words with the same lexical representation will not be able to discriminate between these two words acoustically and must rely on the language model to discriminate between them.

[1]"Cohort size" is used here (as with Shipman and Zue (1982)) to mean the number of distinct vocabulary items that match a particular broad-class encoding. It is not intended to imply a particular theory of lexical access.

Before proceeding, it may useful to set upper and lower bounds for this exercise (Table 1). An obvious upper bound is the full phonemic disambiguation of every word. Of the 6247 words in the dictionary, 6152 unique pronunciations are found (a few cohorts consisting of sets of homophones). A convenient lower bound is the lexical stress pattern of the word, devoid of any segmental information: e.g., "unidirectional" has its stress on the 4th of 6 syllables; hence, 000100 is its lexical-stress profile. 84 unique lexical-stress profiles exist in the dictionary.

Between these two bounds, three variant broad-class partitions were explored for isolated word recognition. All three use the lower-bound stress profile as a starting place, combined with full phonemic information for the syllable with primary stress. The first, with no additional segmental information, produces 4709 distinct cohorts. The second adds a consonant-vowel (CV) profile for the unstressed syllables, which boosts the number of distinct cohorts to 5609. The final partition replaces the CV profile with a six-class manner-based broad-class partition (Nasals, Stops, Fricatives, Glides, Liquids, and Vowels). Including a manner-class representation for unstressed vowels increases the number of cohorts to 6076, which is very close to the upper bound. Thus, there is not much loss of lexical discriminability when using this type of representation.

3.1 Caveats

Now, for this scheme to be maximally useful for recognition, several conditions must obtain. First, we have assumed that we can reliably detect lexically stressed syllables within the speech signal. Waibel (Waibel, 1988) has shown that stress correlates with various acoustic cues such as spectral change. As a side experiment, we have shown that very basic methods provide encouraging results (only sketched here due to space constraints). We re-annotated TIMIT with lexical stress markings, where all frames of each stressed syllable (including onset and coda consonants, not just the nucleus) were marked as stressed. A multi-layer perceptron with 100 hidden units was trained to predict $P(\mathrm{Stress}|\mathrm{Acoustics})$ with a nine-frame context window. No additional phonetic information besides the binary label stressed/unstressed was used in training. Frame-by-frame results on the TIMIT test set were 75% accurate (chance: 52%), and when

MLP output was greater than 0.9, a precision of 89% was obtained (recall: 20%). While far from perfect, this result strongly suggests that even very simple methods can predict lexical stress fairly reasonably.

A second assumption in the above analysis was that words occur in isolation. It is clear that in connected speech, there are a larger number of potential lexical confusions. A third assumption is that those features we are relying upon in our partitions (namely, all features within stressed syllables, and manner of articulation for unstressed syllables) are perfectly reliable and discriminable. In the next two sections, we relax these assumptions by applying extensions of this method to connected speech.

4 Experiment 2: What does a hybrid representation buy you?

As Experiment 1 shows, the hybrid phone/feature representation does not drastically decrease the discriminability of the (albeit small) lexicon. It is also possible that such a representation reduces pronunciation variation, by allowing the canonical representation to more closely match actual pronunciations. For example, we have demonstrated that for common ASR corpora (Switchboard and TIMIT), segments in unstressed syllables were much more likely to deviate from their canonical lexical representation (Fosler-Lussier et al., 1999). If phones that deviate from canonical still keep the same manner class, then a dictionary built with Briscoe-esque representations should more closely match the actual pronunciations of words in running speech (as transcribed by a phonetician).

4.1 Method

In order to test this theory, we used phonetic data from (Fosler-Lussier et al., 1999) in which the ICSI phonetic transcripts of the Switchboard corpus (Greenberg et al., 1996; NIST, 1992) were aligned to a syllabified version of the Pronlex dictionary (Linguistic Data Consortium (LDC), 1996), which has 71014 entries for 66293 words. In this alignment, for every canonical phone given by the lexicon, there were zero or more corresponding realized phones. From these data we extracted the canonical and realized pronunciation of each word token, for a total of 38,527 tokens. Generally, high-frequency function words show the most variation, so they may benefit most from a manner-based representation.

Lexicon type	Strict matching	Matching w/ deletion
1) Phonetic units	37.0%	50.1%
2) Manner-based function words	50.2%	69.6%
3) + Manner for unstressed syls	53.4%	74.6%
4) + Manner for secondary stress	55.7%	77.9%
5) Manner for all syls	60.7%	85.2%

Table 2: Percent of words pronounced canonically for phonetic and hybrid lexical representations

Given these word pronunciation data, we can examine how many word tokens have transcriptions that match their dictionary-listed pronunciations, given the broad-class mappings for various sets of syllables. We built lexica and mapped phonetic transcriptions according to five different criteria:

1. Every segment is phone based (no classes).
2. Function words use manner-based classes.
3. Unstressed syllables and function words use manner only.
4. Secondary stressed syllables also use manner. (Primary stressed syllables are phone based.)
5. Every segment uses manner-based classes.

We noted in the data (as others have done) that a large proportion of the pronunciation variation was due to phone deletion (29% of words) — which would not be handled by the manner-based lexicon. However, it is likely that not every phone deletion leads to an ASR error (as attested by the fact that state-of-the-art Switchboard ASR error rates are typically less than 29%). Often there is enough residual phonetic evidence of the deleted phone, or enough phonetic evidence in other parts of the word, to recognize a word correctly despite the deletion. Thus, we decided to use a two-part strategy in calculating canonical pronunciation (Table 2). The first column, "strict matching", allows no insertions or deletions when comparing the canonical and realized pronunciation. "Matching with deletion" reports the ideal situation where phone deletions were perfectly recoverable in their canonical form. Including and ignoring deletions provides upper and lower bounds on the true lexical access results. (Insertions are relatively rare and not anticipated to affect the results significantly, and hence are not examined.)

4.2 Results and Discussion

In Table 2, we see that a standard ASR lexicon approach (strict matching 1), does not match the transcribed data very well, with only 37% of words pronounced according to the dictionary. The strict matching hybrid scenario on line 3 most closely resembles Briscoe's experiment, and shows a marked improvement in matching the dictionary and realized pronunciations; comparing the two, we see that using manner-based broad classes reduces mismatch by 25% of the total error (from 63% error to 47%), most of which comes from improved modeling of function words (line 2). Whether this gain in representation is worthwhile will depend of course on the cost in terms of the increased hypothesis space.

By allowing for perfect deletion recovery (which will of necessity entail another large expansion of the hypothesis space), a somewhat more optimistic is obtained. Comparing the "matching with deletion" columns of lines 1 and 3, we see that a little over half of the non-deletion pronunciation variation is due to manner changes in unstressed syllables. Again, a good chunk of this is in function words. By moving to manner class for stressed syllables as well would bring the hypothetical error from 25% to 15%, but at the cost of a huge explosion in the hypothesis space (as Briscoe rightly points out and as discussed in the next section).

One interesting implication of this data is that over all types of segments (stressed and unstressed), roughly three-quarters of word pronunciation variants differ from the canonical only in terms of within-manner variation and phonetic deletion.

The moral of this story is that manner-based broad classes may be a useful type of back off from truly reduced and variable syllables (particularly function words), but the full benefit of such a maneuver would only be realized after a reasonable solution for recovering large-scale deletions is found. This may come from predicting with increased specificity where deletions are likely to occur (e.g., complex codas), and what reduced realizations (e.g., of function words) are most common.

5 Experiment 3: What is the cost of a hybrid representation?

Briscoe measured the cost of hybrid representation in terms of the number of lexical activations that a partially-completed word creates (see Section 2). Yet Briscoe's methodology has several shortcomings when applied to today's ASR technology; a summary of the arguments presented above are: (1) Tree-based lexica now share processing for words with identical prefixes. (2) New words are activated only when other word hypotheses end. (3) We now have a large amount of phonetically transcribed, spontaneous speech. (4) Perfect stress detection is not really achievable.

Given criticism 1, a better measure of potential processing requirements is to generate a lattice of hypothesized words and count the number of arcs in the lattice. This lattice can be constructed in such a way that criticism 2 is satisfied. In the next section, we present a finite state machine formalism for generating such a lattice.

We apply this technique to the phonetic transcription of the Switchboard corpus (thus alleviating criticism 3). However, this introduces several problems. As Experiment 2 shows, many words have pronunciations that do not appear in the dictionary. Thus, we must find a way to alleviate the mismatch between the phonetic transcription and the dictionary in a way that is plausible for ASR processing.

We can address criticism 4 by creating phone-based and manner-based transcriptions that will run in parallel; thus, the lattice generator would be free to choose whichever representation allows the matching to a dictionary word.

5.1 Method

In this experiment we consider a finite-state transducer model of the strategy described above. This corresponds not to the ASR system as a whole, but rather to the pronunciation model of a traditional system. We assume that the pronunciation as given by the transcriber is correct, but we model the transformation of realized phones into canonical dictionary pronunciations. Since we are only investigating the combined acoustic-phonetic-lexical representation, we have left out the re-weighting and pruning of hypotheses due to integration of a language model, discourse model, or any other constraints.

Specifically, this model consists of three finite state transducers composed. The first FSM, **R**, encodes the representation of the **r**ealized phonetic transcription of the spoken corpus. In order to match this to dictionary pronunciations, we train a confusion matrix on all realized/canonical phone pairs, to obtain $P(\text{dictionary phone}|\text{transcribed phone})$; these confusion probabilities are encoded as a finite state transducer **C**. Thus, **C** is derived by computing the strength of all correspondences between the phonetic transcription of what was actually said at the phone level and the canonical pronunciation of the corresponding words. This confusion matrix consists of three parts, corresponding to substitutions, insertions, and deletions.

1. Pairwise substitutions are counted to yield a standard confusion matrix.

2. Where two or more realized phones correspond to a single canonical phone (a rare occurrence, as in e.g., *really* /r iy l iy/ → [r ih ax l iy]), each realized phone is allowed (independently) to be either deleted or substituted with its pairwise confusions from (1).

3. Deleted phones are assumed to be potentially recoverable (as in Experiment 2), so both an epsilon transition and the canonical pronunciation are preserved in the confusion matrix.

In each of these confusion matrices, we have always preserved the pathway from each realized utterance to its canonical representation for the whole corpus. So for this seen corpus, it is always possible in theory to recover the canonical representation, such that the right answer is always one of the possible hypotheses. While this may seem a bit strange, here we can only overestimate the potential hypothesis space (by adding the correct string and by assuming that deletions are recoverable); the point of this exercise is to see the number of total hypotheses (the search space) generated under such a system.

The third transducer, **D**, is the ASR dictionary that we wish to test. Thus, composing $\mathbf{R} \circ \mathbf{C} \circ \mathbf{D}$ will give the graph of all potential complete hypotheses in this space. Figure 1 shows a pruned hypothesis graph for the phrase "it's really sad" (the full hypothesis graph has 12216 arcs).

5.2 Results and Discussion

By choosing different sub-word representations, we can test Briscoe's contention that backing off to manner-based broad classes for certain (e.g., unstressed) syllables will reduce the search space and/or facilitate recovery of the intended word string. When a phone is substituted with a manner class, we construct **C** so that the generated confusions are over manner classes rather than phones.

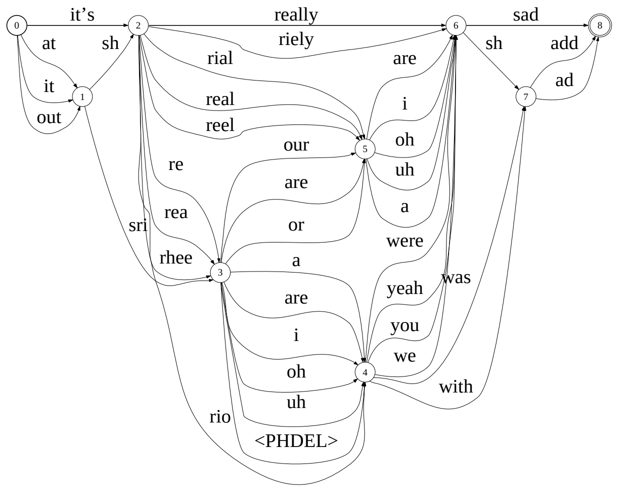


Figure 1: Pruned hypothesis graph for *It's really sad*

Figure 2 shows how the number of hypotheses per word changes as a function of the number of words in the hypothesis. Note that if the relationship were linear, we would expect to see a flat line. The figure demonstrates that that Briscoe's conclusions were correct, given the assumption that one can accurately detect lexical stress (as illustrated by the line with circles on 2). Across all utterances, the average number of hypotheses per word for the hybrid dictionary was 510 (roughly 1/3 of the phone-based average of 1429). However, when one allows for the fact that stress detection is not perfect, one sees an *increase* in the amount of necessary computation: the non-ideal hybrid dictionary has an average of 3322

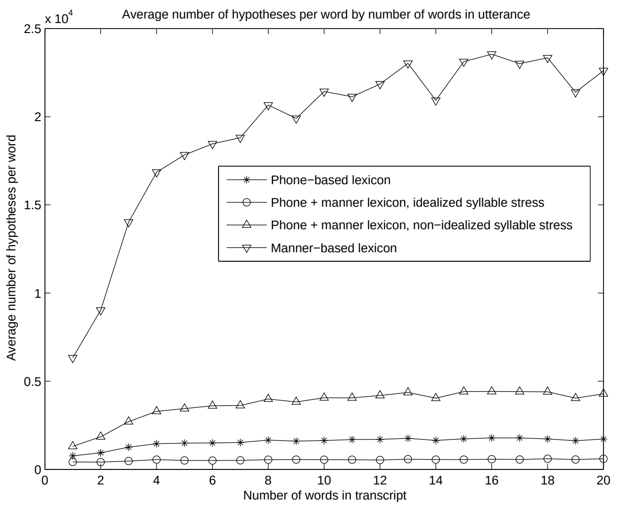


Figure 2: Average number of hypotheses per word as a function of number of words in utterance

hypotheses per word (2.3 times the phone-based average). Yet this is much lower than the potential growth of the hypothesis space given with manner-only dictionaries. This dictionary generated a hypothesis space 12 times as large as a phone based dictionary (17186 hypotheses/word average); moreover, the curve grows significantly as a function of the number of words, so longer utterances will take disproportionately more space. Thus, Briscoe's hypothesis that purely manner-based decoding is too expensive seems to be confirmed.

6 Integration into ASR

This paper has investigated hybrid representations along computational phonology lines, but we have also trained an ASR system with a hybrid lexicon for the Wall Street Journal (WSJ0) corpus. Space does not permit a full explanation of the experiment here (for more details, see (Fosler-Lussier et al., 2005)), but we include the results from this experiment as evidence of the validity of the approach.

In this experiment, we trained phonetic and manner-based acoustic models for all segments using the flat-start recipe of the HTK recognizer (Young et al., 2002). After a number of iterations of EM-training, we constructed a hybrid set of acoustic models and lexicon in which phones in unstressed syllables were replaced with manner classes (Hybrid-all). We also derived a lexicon in which the recognizer could choose whether a manner or phonetic representation was appropriate for unstressed segments (Hybrid-choice). During evaluation, we found that the Hybrid-choice lexicon degraded only slightly over a phone-based lexicon (9.9% word error vs. 9.1%), and in fact improved recognition in mild (10dB SNR) additive car noise (13.0% vs. 15.4%). The Hybrid-all was worse on clean speech (13.1% WER) but statistically the same as phone-based on noisy speech (15.8%). While not conclusive, this suggests that hybrid models may provide an interesting avenue for robustness research.

7 Conclusion

Our studies verify to some degree Briscoe's claim that a hybrid representation for lexical modeling, with stressed syllables receiving full phonetic representation and unstressed syllables represented by manner classes, can improve ASR processing. However, our analysis shows that the argument for this hypothesis plays out along very different lines than in Briscoe's study. A hybrid phone-manner lexicon can theoretically benefit ASR because (a) the discriminative power of the lexicon is not reduced greatly, (b) such a representation is a much better model of the types of pronunciation variation seen in spontaneous speech corpora such as Switchboard, and (c) the theoretical average hypothesis space increases only by a little over a factor of 2. This last fact is contrary to Briscoe's finding that the search space would be reduced because it incorporates more realistic assumptions about the detection of stressed versus unstressed syllables.

These experiments were designed primarily to investigate the validity of Briscoe's claims, and thus we attempted to remain true to his model. However, it is clear that our analysis can be extended in several ways. We have begun experimenting with pruning the hypothesis graph to remove unlikely arcs – this would give a more accurate model of the ASR processing that would occur. However, this only makes sense if language model constraints are integrated into the processing, since some word sequences in the graph would be discarded as unlikely. This analysis could also benefit from a more accurate model of the ASR system's transformation between realized phones and lexical representations. This could be achieved by comparing the Gaussian acoustic model distributions in an HMM system or sampling the acoustic model's space (McAllaster et al., 1998). Both of these extensions will be considered in future work.

The results clearly indicate that further investigation and development of a hybrid lexical strategy in an ASR system is worthwhile, particularly for spontaneous speech corpora where the problem of pronunciation variation is most rampant.

Acknowledgments

The authors would like to thank Keith Johnson, Monica Rajamanohar, and Yu Wang for discussion of this work. This work was funded in part by NSF grant ITR-0427413; the opinions and conclusions expressed in this work are those of the authors and not of any funding agency.

References

E. J. Briscoe. 1989. Lexical access in connected speech recognition. In *Proc. 27th Annual Meeting of the Association for Computational Linguistics*, pages 84–90.

R. Carlson, K. Elenius, B. Granström, and H. Hunnicutt. 1985. Phonetic and orthographic properties of the basic vocabulary of five european languages. In *STL-QPSR 1/1985*, pages 63–94, Stockholm. Speech Transmission Laboratory, Dept. of Speech Communication, Royal Institute of Technology.

K. W. Church. 1987. *Phonological Parsing in Speech Recognition*. Kluwer, Dordrecht.

M. Finke, J. Fritsch, and D. Koll. 1999. Modeling and efficient decoding of large vocabulary conversational speech. In *Conversational Speech Recognition Workshop: DARPA Hub-5E Evaluation*.

W. Fisher, 1996. *The tsylb2 Program: Algorithm Description*. NIST. Part of the tsylb2-1.1 package.

E. Fosler-Lussier, S. Greenberg, and N. Morgan. 1999. Incorporating contextual phonetics into automatic speech recognition. In *Int'l Congress of Phonetic Sciences*, San Francisco, California.

E. Fosler-Lussier, C. A. Rytting, and S. Srinivasan. 2005. Phonetic ignorance is bliss: Investigating the effects of phonetic information reduction on asr performance. In *Proc. Interspeech*, Lisbon, Portugal.

J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren. 1993. DARPA TIMIT acoustic-phonetic continuous speech corpus. Technical Report NISTIR 4930, NIST, Gaithersburg, MD.

S. Greenberg, J. Hollenbach, and D. Ellis. 1996. Insights into spoken language gleaned from phonetic transcription of the switchboard corpus. In *Proc. 4th Int'l Conference on Spoken Language Processing*. Philadelphia, PA.

R. Jakobson, G. Fant, and M. Halle. 1952. Preliminaries to speech analysis. Technical Report 13, Acoustics Laboratory, Massachusetts Instutite of Technology.

A. Juneja and C. Espy-Wilson. 2004. Significance of invariant acoustic cues in a probabilistic framework for landmark-based speech recognition. In *From Sound to Sense: Fifty+ Years of Discoveries in Speech Communication*, Cambridge, MA. MIT.

K. Kirchhoff. 1998. Combining articulatory and acoustic information for speech recognition in noisy and reverberant environments. In *Proc. 5th Int'l Conference on Spoken Language Processing*, Sydney.

Linguistic Data Consortium (LDC). 1996. The PRONLEX pronunciation dictionary. Available from the LDC, ldc@unagi.cis.upenn.edu. Part of the COMLEX distribution.

K. Livescu, J. Glass, and J. Bilmes. 2003. Hidden feature models for speech recognition using dynamic bayesian networks. In *Proc. 8th European Conference on Speech Communication and Technology*, Geneva, Switzerland.

D. McAllaster, L. Gillick, F. Scattone, and M. Newman. 1998. Fabricating conversational speech data with acoustic models: A program to examine model-data mismatch. In *Proc. 5th Int'l Conference on Spoken Language Processing*, pages 1847–1850, Sydney, Australia.

G. Miller and P. Nicely. 1955. Analysis of some perceptual confusions among some english consonants. *Journal of Acoustical Society of America*, 27:338–52.

M. Mohri, F. Pereira, and M. Riley, 2001. *AT&T FSM Library$^{\mathrm{TM}}$ – General-Purpose Finite-State Machine Software Tools*. AT&T, Florham Park, New Jersey. Available at http://www.research.att.com/sw/tools/fsm.

NIST. 1992. Switchboard Corpus: Recorded telephone conversations. National Institute of Standards and Technology Speech Disc 9-1 to 9-25.

M. Ostendorf. 1999. Moving beyound the 'beads-on-a-string' model of speech. In *1999 IEEE Workshop on Automatic Speech Recognition and Understanding*, Keystone, Colorado.

M. Saraçlar, H. Nock, and S. Khudanpur. 2000. Pronunciation modeling by sharing Gaussian densities across phonetic models. *Computer Speech and Language*, 14:137–160.

D. W. Shipman and V. W. Zue. 1982. Properties of large lexicons: Implications for advanced isolated word recognition systems. In *Proc. Int'l Conference on Acoustics, Speech, and Signal Processing*, volume 82, pages 546–549, Paris, France.

K. Stevens. 1981. Invariant acoustic correlates of phonetic features. *Journal of Acoustical Society of America*, 69 suppl. 1:S31.

A. Waibel. 1988. *Prosody and Speech Recognition*. Morgan Kaufmann, San Mateo, California.

S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, 2002. *The HTK Book*. Cambridge Unveristy Engineering Department. http://htk.eng.cam.ac.uk.

Emotions from text: machine learning for text-based emotion prediction

Cecilia Ovesdotter Alm[*]
Dept. of Linguistics
UIUC
Illinois, USA
ebbaalm@uiuc.edu

Dan Roth
Dept. of Computer Science
UIUC
Illinois, USA
danr@uiuc.edu

Richard Sproat
Dept. of Linguistics
Dept. of Electrical Eng.
UIUC
Illinois, USA
rws@uiuc.edu

Abstract

In addition to information, text contains attitudinal, and more specifically, emotional content. This paper explores the *text-based emotion prediction problem* empirically, using supervised machine learning with the SNoW learning architecture. The goal is to classify the emotional affinity of sentences in the narrative domain of children's fairy tales, for subsequent usage in appropriate expressive rendering of text-to-speech synthesis. Initial experiments on a preliminary data set of 22 fairy tales show encouraging results over a naïve baseline and BOW approach for classification of emotional versus non-emotional contents, with some dependency on parameter tuning. We also discuss results for a tripartite model which covers emotional valence, as well as feature set alternations. In addition, we present plans for a more cognitively sound sequential model, taking into consideration a larger set of basic emotions.

1 Introduction

Text does not only communicate informative contents, but also attitudinal information, including emotional states. The following reports on an empirical study of *text-based emotion prediction*.

Section 2 gives a brief overview of the intended application area, whereas section 3 summarizes related work. Next, section 4 explains the empirical study, including the machine learning model, the corpus, the feature set, parameter tuning, etc. Section 5 presents experimental results from two classification tasks and feature set modifications. Section 6 describes the agenda for refining the model, before presenting concluding remarks in 7.

2 Application area: Text-to-speech

Narrative text is often especially prone to having emotional contents. In the literary genre of fairy tales, emotions such as HAPPINESS and ANGER and related cognitive states, e.g. LOVE or HATE, become integral parts of the story plot, and thus are of particular importance. Moreover, the story teller reading the story interprets emotions in order to orally convey the story in a fashion which makes the story come alive and catches the listeners' attention.

In speech, speakers effectively express emotions by modifying prosody, including pitch, intensity, and durational cues in the speech signal. Thus, in order to make text-to-speech synthesis sound as natural and engaging as possible, it is important to convey the emotional stance in the text. However, this implies first having identified the appropriate emotional meaning of the corresponding text passage.

Thus, an application for emotional text-to-speech synthesis has to solve two basic problems. First, what emotion or emotions most appropriately describe a certain text passage, and second, given a text passage and a specified emotional mark-up, how to render the prosodic contour in order to convey the emotional content, (Cahn, 1990). The *text-based emotion prediction* task (TEP) addresses the first of these two problems.

3 Previous work

For a complete general overview of the field of *affective computing*, see (Picard, 1997). (Liu, Lieberman and Selker, 2003) is a rare study in text-based inference of sentence-level emotional affinity. The authors adopt the notion of *basic emotions*, cf. (Ekman, 1993), and use six emotion categories: ANGER, DISGUST, FEAR, HAPPINESS, SADNESS, SURPRISE. They critique statistical NLP for being unsuccessful at the small sentence level, and instead use a database of common-sense knowledge and create affect models which are combined to form a representation of the emotional affinity of a sentence. At its core, the approach remains dependent on an emotion lexicon and hand-crafted rules for conceptual polarity. In order to be effective, emotion recognition must go beyond such resources; the authors note themselves that lexical affinity is fragile. The method was tested on 20 users' preferences for an email-client, based on user-composed text emails describing short but colorful events. While the users preferred the emotional client, this evaluation does not reveal emotion classification accuracy, nor how well the model generalizes on a large data set.

Whereas work on emotion classification from the point of view of natural speech and human-computer dialogues is fairly extensive, e.g. (Scherer, 2003), (Litman and Forbes-Riley, 2004), this appears not to be the case for text-to-speech synthesis (TTS). A short study by (Sugimoto et al., 2004) addresses sentence-level emotion recognition for Japanese TTS. Their model uses a composition assumption: the emotion of a sentence is a function of the emotional affinity of the words in the sentence. They obtain emotional judgements of 73 adjectives and a set of sentences from 15 human subjects and compute words' emotional strength based on the ratio of times a word or a sentence was judged to fall into a particular emotion bucket, given the number of human subjects. Additionally, they conducted an interactive experiment concerning the acoustic rendering of emotion, using manual tuning of prosodic parameters for Japanese sentences. While the authors actually address the two fundamental problems of emotional TTS, their approach is impractical and most likely cannot scale up for a real corpus. Again, while lexical items with clear emotional meaning, such as *happy* or *sad*, matter, emotion classification probably needs to consider additional inference mechanisms. Moreover, a naïve compositional approach to emotion recognition is risky due to simple linguistic facts, such as context-dependent semantics, domination of words with multiple meanings, and emotional negation.

Many NLP problems address attitudinal meaning distinctions in text, e.g. detecting *subjective* opinion documents or expressions, e.g. (Wiebe et al, 2004), measuring *strength* of subjective clauses (Wilson, Wiebe and Hwa, 2004), determining word *polarity* (Hatzivassiloglou and McKeown, 1997) or texts' attitudinal valence, e.g. (Turney, 2002), (Bai, Padman and Airoldi, 2004), (Beineke, Hastie and Vaithyanathan, 2003), (Mullen and Collier, 2003), (Pang and Lee, 2003). Here, it suffices to say that the targets, the domain, and the intended application differ; our goal is to classify emotional text passages in children's stories, and eventually use this information for rendering expressive child-directed storytelling in a text-to-speech application. This can be useful, e.g. in therapeutic education of children with communication disorders (van Santen et al., 2003).

4 Empirical study

This part covers the experimental study with a formal problem definition, computational implementation, data, features, and a note on parameter tuning.

4.1 Machine learning model

Determining emotion of a linguistic unit can be cast as a multi-class classification problem. For the flat case, let T denote the text, and s an embedded linguistic unit, such as a sentence, where $s \in T$. Let k be the number of emotion classes $E = \{em_1, em_2, .., em_k\}$, where em_1 denotes the special case of *neutrality*, or absence of emotion. The goal is to determine a mapping function $f : s \rightarrow em_i$, such that we obtain an ordered labeled pair (s, em_i). The mapping is based on $F = \{f_1, f_2, .., f_n\}$, where F contains the features derived from the text.

Furthermore, if multiple emotion classes can characterize s, then given $E' \subset E$, the target of the mapping function becomes the ordered pair (s, E'). Finally, as further discussed in section 6, the hierarchical case of label assignment requires a sequential model that further defines levels of coarse versus fine-grained classifiers, as done by (Li and Roth, 2002) for the *question classification* problem.

4.2 Implementation

Whereas our goal is to predict finer emotional meaning distinctions according to emotional categories in speech; in this study, we focus on the basic task of recognizing emotional passages and on determining their valence (i.e. positive versus negative) because we currently do not have enough training data to explore finer-grained distinctions. The goal here is to get a good understanding of the nature of the TEP problem and explore features which may be useful.

We explore two cases of flat classification, using a variation of the Winnow update rule implemented in the SNoW learning architecture (Carlson et al., 1999),[1] which learns a linear classifier in feature space, and has been successful in several NLP applications, e.g. semantic role labeling (Koomen, Punyakanok, Roth and Yih, 2005). In the first case, the set of emotion classes E consists of EMOTIONAL versus non-emotional or NEUTRAL, i.e. $E = \{N, E\}$. In the second case, E has been incremented with emotional distinctions according to the valence, i.e. $E = \{N, PE, NE\}$. Experiments used 10-fold cross-validation, with 90% train and 10% test data.[2]

4.3 Data

The goal of our current data annotation project is to annotate a corpus of approximately 185 children stories, including Grimms', H.C. Andersen's and B. Potter's stories. So far, the annotation process proceeds as follows: annotators work in pairs on the same stories. They have been trained separately and work independently in order to avoid any annotation bias and get a true understanding of the task difficulty. Each annotator marks the sentence level with one of eight *primary emotions*, see table 1, reflecting an extended set of *basic emotions* (Ekman, 1993). In order to make the annotation process more focused, emotion is annotated from the point of view of the text, i.e. the *feeler* in the sentence. While the primary emotions are targets, the sentences are also marked for other affective contents, i.e. background *mood*, *secondary* emotions via *intensity*, *feeler*, and *textual* cues. Disagreements in annotations are resolved by a second pass of tie-breaking by the first author, who chooses one of the competing labels. Eventually, the completed annotations will be made available.

Table 1: Basic emotions used in annotation

Abbreviation	Emotion class
A	ANGRY
D	DISGUSTED
F	FEARFUL
H	HAPPY
Sa	SAD
Su+	POSITIVELY SURPRISED
Su-	NEGATIVELY SURPRISED

Emotion annotation is hard; interannotator agreement currently range at $\kappa = .24 - .51$, with the ratio of observed annotation overlap ranging between 45-64%, depending on annotator pair and stories assigned. This is expected, given the subjective nature of the annotation task. The lack of a clear definition for emotion vs. non-emotion is acknowledged across the emotion literature, and contributes to dynamic and shifting annotation targets. Indeed, a common source of confusion is NEUTRAL, i.e. deciding whether or not a sentence is emotional or non-emotional. Emotion perception also depends on which character's point-of-view the annotator takes, and on extratextual factors such as annotator's personality or mood. It is possible that by focusing more on the training of annotator pairs, particularly on joint training, agreement might improve. However, that would also result in a bias, which is probably not preferable to actual perception. Moreover, what agreement levels are needed for successful expressive TTS remains an empirical question.

The current data set consisted of a preliminary annotated and tie-broken data set of 1580 sentence, or 22 Grimms' tales. The label distribution is in table 2. NEUTRAL was most frequent with 59.94%.

Table 2: Percent of annotated labels

A	D	F	H
12.34%	0.89%	7.03%	6.77%
N	SA	SU+	SU.-
59.94%	7.34%	2.59%	3.10%

[1] Available from http://l2r.cs.uiuc.edu/∼cogcomp/

[2] Experiments were also run for Perceptron, however the results are not included. Overall, Perceptron performed worse.

Table 3: % EMOTIONAL vs. NEUTRAL examples

E	N
40.06%	59.94%

Table 4: % POSITIVE vs. NEGATIVE vs. NEUTRAL

PE	NE	N
9.87%	30.19%	59.94%

Next, for the purpose of this study, all emotional classes, i.e. A, D, F, H, SA, SU+, SU-, were combined into one emotional superclass E for the first experiment, as shown in table 3. For the second experiment, we used two emotional classes, i.e. positive versus negative emotions; PE={H, SU+} and NE={A, D, F, SA, SU-}, as seen in table 4.

4.4 Feature set

The feature extraction was written in python. SNoW only requires active features as input, which resulted in a typical feature vector size of around 30 features. The features are listed below. They were implemented as boolean values, with continuous values represented by ranges. The ranges generally overlapped, in order to get more generalization coverage.

1. First sentence in story
2. Conjunctions of selected features (see below)
3. Direct speech (i.e. whole quote) in sentence
4. Thematic story type (3 top and 15 sub-types)
5. Special punctuation (! and ?)
6. Complete upper-case word
7. Sentence length in words (0-1, 2-3, 4-8, 9-15, 16-25, 26-35, $>$35)
8. Ranges of story progress (5-100%, 15-100%, 80-100%, 90-100%)
9. Percent of JJ, N, V, RB (0%, 1-100%, 50-100%, 80-100%)
10. V count in sentence, excluding participles (0-1, 0-3, 0-5, 0-7, 0-9, $>$ 9)
11. Positive and negative word counts ($\geq$ 1, $\geq$ 2, $\geq$ 3, $\geq$ 4, $\geq$ 5, $\geq$ 6)
12. WordNet emotion words
13. Interjections and affective words
14. *Content BOW*: N, V, JJ, RB words by POS

Feature conjunctions covered pairings of counts of positive and negative words with range of story progress or interjections, respectively.

Feature groups 1, 3, 5, 6, 7, 8, 9, 10 and 14 are extracted automatically from the sentences in the stories; with the SNoW POS-tagger used for features 9, 10, and 14. Group 10 reflects how many verbs are active in a sentence. Together with the quotation and punctuation, verb domination intends to capture the assumption that emotion is often accompanied by increased action and interaction. Feature group 4 is based on Finish scholar Antti Aarne's classes of folk-tale types according to their informative thematic contents (Aarne, 1964). The current tales have 3 top story types (ANIMAL TALES, ORDINARY FOLK-TALES, and JOKES AND ANECDOTES), and 15 subtypes (e.g. *supernatural helpers* is a subtype of the ORDINARY FOLK-TALE). This feature intends to provide an idea about the story's general affective *personality* (Picard, 1997), whereas the feature reflecting the story progress is hoped to capture that some emotions may be more prevalent in certain sections of the story (e.g. the happy end).

For semantic tasks, words are obviously important. In addition to considering 'content words', we also explored specific word lists. Group 11 uses 2 lists of 1636 positive and 2008 negative words, obtained from (Di Cicco et al., online). Group 12 uses lexical lists extracted from WordNet (Fellbaum, 1998), on the basis of the primary emotion words in their adjectival and nominal forms. For the adjectives, Py-WordNet's (Steele et al., 2004) SIMILAR feature was used to retrieve similar items of the primary emotion adjectives, exploring one additional level in the hierarchy (i.e. similar items of all senses of all words in the synset). For the nouns and any identical verbal homonyms, synonyms and hyponyms were extracted manually.[3] Feature group 13 used a short list of 22 interjections collected manually by browsing educational ESL sites, whereas the affective word list of 771 words consisted of a combination of the non-neutral words from (Johnson-Laird and Oatley, 1989) and (Siegle, online). Only a subset of these lexical lists actually occurred.[4]

[3]Multi-words were transformed to hyphenated form.

[4]At this point, neither stems and bigrams nor a list of onomatopoeic words contribute to accuracy. Intermediate resource processing inserted some feature noise.

The above feature set is henceforth referred to as *all features*, whereas *content BOW* is just group 14. The *content BOW* is a more interesting baseline than the naïve one, *P(Neutral)*, i.e. always assigning the most likely NEUTRAL category. Lastly, emotions blend and transform (Liu, Lieberman and Selker, 2003). Thus, emotion and background mood of immediately adjacent sentences, i.e. the *sequencing*, seems important. At this point, it is not implemented automatically. Instead, it was extracted from the manual emotion and mood annotations. If *sequencing* seemed important, an automatic method using sequential target activation could be added next.

4.5 Parameter tuning

The Winnow parameters that were tuned included promotional α, demotional β, activation threshold θ, initial weights ω, and the regularization parameter, S, which implements a margin between positive and negative examples. Given the currently fairly limited data, results from 2 alternative tuning methods, applied to *all features*, are reported.

- For the condition called *sep-tune-eval*, 50% of the sentences were randomly selected and set aside to be used for the parameter tuning process only. Of this subset, 10% were subsequently randomly chosen as test set with the remaining 90% used for training during the automatic tuning process, which covered 4356 different parameter combinations. Resulting parameters were: $\alpha = 1.1$, $\beta = 0.5$, $\theta = 5$, $\omega = 1.0$, $S = 0.5$. The remaining half of the data was used for training and testing in the 10-fold cross-validation evaluation. (Also, note the slight change for *P(Neutral)* in table 5, due to randomly splitting the data.)
- Given that the data set is currently small, for the condition named *same-tune-eval*, tuning was performed automatically on all data using a slightly smaller set of combinations, and then manually adjusted against the 10-fold cross-validation process. Resulting parameters were: $\alpha = 1.2$, $\beta = 0.9$, $\theta = 4$, $\omega = 1$, $S = 0.5$. All data was used for evaluation.

Emotion classification was sensitive to the selected tuning data. Generally, a smaller tuning set resulted in pejorative parameter settings. The random selection could make a difference, but was not explored.

5 Results and discussion

This section first presents the results from experiments with the two different confusion sets described above, as well as feature experimentation.

5.1 Classification results

Average accuracy from 10-fold cross validation for the first experiment, i.e. classifying sentences as either NEUTRAL or EMOTIONAL, are included in table 5 and figure 1 for the two tuning conditions on the main feature sets and baselines. As expected,

Table 5: Mean classification accuracy: N vs. E, 2 conditions

	same-tune-eval	sep-tune-eval
P(Neutral)	59.94	60.05
Content BOW	61.01	58.30
All features except BOW	64.68	63.45
All features	68.99	63.31
All features + sequencing	69.37	62.94

degree of success reflects parameter settings, both for *content BOW* and *all features*. Nevertheless, under these circumstances, performance above a naïve baseline and a BOW approach is obtained. Moreover, *sequencing* shows potential for contributing in one case. However, observations also point to three issues: first, the current data set appears to be too small. Second, the data is not easily separable. This comes as no surprise, given the subjective nature of the task, and the rather low interannotator agreement, reported above. Moreover, despite the schematic narrative plots of children's stories, tales still differ in their overall affective orientation, which increases data complexity. Third and finally, the EMOTION class is combined by basic emotion labels, rather than an original annotated label.

More detailed averaged results from 10-fold cross-validation are included in table 6 using *all features* and the separated tuning and evaluation data condition *sep-tune-eval*. With these parameters, approximately 3% improvement in accuracy over the naïve baseline *P(Neutral)* was recorded, and 5% over the *content BOW*, which obviously did poorly with these parameters. Moreover, precision is

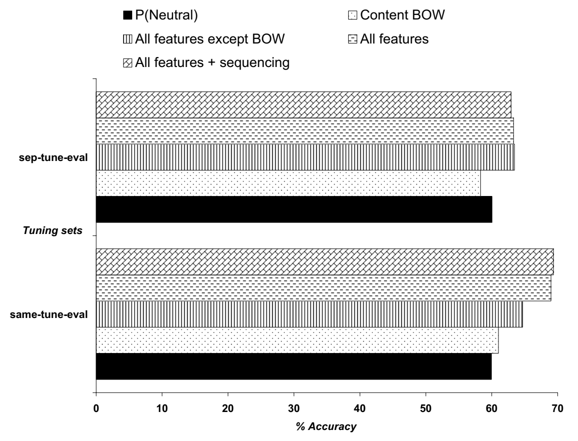


Figure 1: Accuracy under different conditions (in %)

Table 6: Classifying N vs. E (*all features*, *sep-tune-eval*)

Measure	N	E
Averaged accuracy	0.63	0.63
Averaged error	0.37	0.37
Averaged precision	0.66	0.56
Averaged recall	0.75	0.42
Averaged F-score	0.70	0.47

higher than recall for the combined EMOTION class. In comparison, with the *same-tune-eval* procedure, the accuracy improved by approximately 9% over *P(Neutral)* and by 8% over *content BOW*.

In the second experiment, the emotion category was split into two classes: emotions with positive versus negative valence. The results in terms of precision, recall, and F-score are included in table 7, using *all features* and the *sep-tune-eval* condition. The decrease in performance for the emotion classes mirrors the smaller amounts of data available for each class. As noted in section 4.3, only 9.87% of the sentences were annotated with a positive emotion, and the results for this class are worse. Thus, performance seems likely to improve as more annotated story data becomes available; at this point, we are experimenting with merely around 12% of the total texts targeted by the data annotation project.

5.2 Feature experiments

Emotions are poorly understood, and it is especially unclear which features may be important for their recognition from text. Thus, we experimented

Table 7: N, PE, and NE (*all features*, *sep-tune-eval*)

	N	NE	PE
Averaged precision	0.64	0.45	0.13
Averaged recall	0.75	0.27	0.19
Averaged F-score	0.69	0.32	0.13

Table 8: Feature group members

Word lists	interj., WordNet, affective lists, pos/neg
Syntactic	length ranges, % POS, V-count ranges
Story-related	% story-progress, 1st sent., story type
Orthographic	punctuation, upper-case words, quote
Conjunctions	Conjunctions with pos/neg
Content BOW	Words (N,V,Adj, Adv)

with different feature configurations. Starting with all features, again using 10-fold cross-validation for the separated tuning-evaluation condition *sep-tune-eval*, one additional feature group was removed until none remained. The feature groups are listed in table 8. Figure 2 on the next page shows the accuracy at each step of the cumulative subtraction process. While some feature groups, e.g. syntactic, appeared less important, the removal order mattered; e.g. if syntactic features were removed first, accuracy decreased. This fact also illustrated that features work together; removing any group degraded performance because features interact and there is no true independence. It was observed that features' contributions were sensitive to parameter tuning. Clearly, further work on developing features which fit the TEP problem is needed.

6 Refining the model

This was a “first pass” of addressing TEP for TTS. At this point, the annotation project is still on-going, and we only had a fairly small data set to draw on. Nevertheless, results indicate that our learning approach benefits emotion recognition. For example, the following instances, also labeled with the same valence by both annotators, were correctly classified both in the binary (N vs. E) and the tripartite polarity task (N, NE, PE), given the separated tuning and evaluation data condition, and using *all features*:

(1a) E/NE: Then he offered the dwarfs money, and prayed and besought them to let him take her away; but they said, ”We will not part with her for all the gold in the world.”

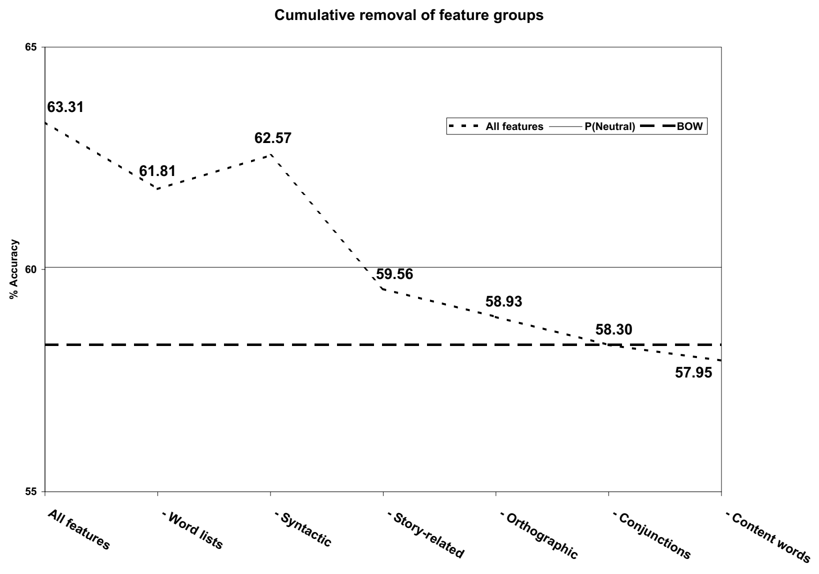


Figure 2: Averaged effect of feature group removal, using *sep-tune-eval*

(1b) N: And so the little girl really did grow up; her skin was as white as snow, her cheeks as rosy as the blood, and her hair as black as ebony; and she was called Snowdrop.

(2a) E/NE: "Ah," she answered, "have I not reason to weep?

(2b) N: Nevertheless, he wished to try him first, and took a stone in his hand and squeezed it together so that water dropped out of it.

Cases (1a) and (1b) are from the well-known FOLK TALE *Snowdrop*, also called *Snow White*. (1a) and (1b) are also correctly classified by the simple *content BOW* approach, although our approach has higher prediction confidence for E/NE (1a); it also considers, e.g. direct speech, a fairly high verb count, advanced story progress, connotative words and conjunctions thereof with story progress features, all of which the BOW misses. In addition, the simple *content BOW* approach makes incorrect predictions at both the bipartite and tripartite levels for examples (2a) and (2b) from the JOKES AND ANECDOTES stories *Clever Hans* and *The Valiant Little Tailor*, while our classifier captures the affective differences by considering, e.g. distinctions in verb count, interjection, POS, sentence length, connotations, story subtype, and conjunctions.

Next, we intend to use a larger data set to conduct a more complete study to establish mature findings. We also plan to explore finer emotional meaning distinctions, by using a hierarchical sequential model which better corresponds to different levels of cognitive difficulty in emotional categorization by humans, and to classify the full set of basic level emotional categories discussed in section 4.3. Sequential modeling of simple classifiers has been successfully employed to question classification, for example by (Li and Roth, 2002). In addition, we are working on refining and improving the feature set, and given more data, tuning can be improved on a sufficiently large development set. The three subcorpora in the annotation project can reveal how authorship affects emotion perception and classification.

Moreover, arousal appears to be an important dimension for emotional prosody (Scherer, 2003), especially in storytelling (Alm and Sproat, 2005). Thus, we are planning on exploring degrees of emotional intensity in a learning scenario, i.e. a problem similar to measuring strength of opinion clauses (Wilson, Wiebe and Hwa, 2004).

Finally, emotions are not discrete objects; rather they have transitional nature, and blend and overlap along the temporal dimension. For example, (Liu, Lieberman and Selker, 2003) include parallel estimations of emotional activity, and include smoothing techniques such as interpolation and decay to capture sequential and interactive emotional activity. Observations from tales indicate that some emotions are more likely to be prolonged than others.

7 Conclusion

This paper has discussed an empirical study of the *text-based emotion prediction* problem in the domain of children's fairy tales, with child-directed expressive text-to-speech synthesis as goal. Besides reporting on encouraging results in a first set of computational experiments using supervised machine learning, we have set forth a research agenda for tackling the TEP problem more comprehensively.

8 Acknowledgments

We are grateful to the annotators, in particular A. Rasmussen and S. Siddiqui. We also thank two anonymous reviewers for comments. This work was funded by NSF under award ITR-#0205731, and NS ITR IIS-0428472. The annotation is supported by UIUC's Research Board. The authors take sole responsibility for the work.

References

Antti Aarne. 1964. *The Types of the Folk-Tale: a Classification and Bibliography.* Helsinki: Suomalainen Tiedeakatemia.

Cecilia O. Alm, and Richard Sproat. 2005. Perceptions of emotions in expressive storytelling. *INTERSPEECH 2005*.

Xue Bai, Rema Padman, and Edoardo Airoldi. 2004. Sentiment extraction from unstructured text using tabu search-enhanced Markov blankets. In *MSW2004*, Seattle.

Philip Beineke, Trevor Hastie, and Shivakumar Vaithyanathan. 2004. The sentimental factor: improving review classification via human-provided information. In *Proceedings of ACL*, 263–270.

Janet Cahn. 1990. The generation of affect in synthesized Speech. *Journal of the American Voice I/O Society*, 8:1–19.

Andrew Carlson, Chad Cumby, Nicholas Rizzolo, Jeff Rosen, and Dan Roth. 1999. *The SNoW Learning Architecture.* Technical Report UIUCDCS-R-99-2101, UIUC Comp. Sci.

Stacey Di Cicco et al. General Inquirer Pos./Neg. lists *http://www.webuse.umd.edu:9090/*

Paul Ekman. 1993. Facial expression and emotion. *American Psychologist*, 48(4), 384–392.

Christiane Fellbaum, Ed. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Vasileios Hatzivassiloglou, and Kathleen McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of ACL*, 174–181.

Philip Johnson-Laird, and Keith Oatley. 1989. The language of emotions: an analysis of a semantic field. *Cognition and Emotion*, 3:81–123.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Annual Conference on Computational Language Learning (CoNLL)*, 181–184.

Diane Litman, and Kate Forbes-Riley. 2004. Predicting student emotions in computer-human tutoring dialogues. In *Proceedings of ACL*, 351–358.

Xin Li, and Dan Roth. 2002. Learning question classifiers: the role of semantic information. In *Proc. International Conference on Computational Linguistics (COLING)*, 556–562.

Hugo Liu, Henry Lieberman, and Ted Selker. 2003. A model of textual affect sensing using real-world knowledge. In *ACM Conference on Intelligent User Interfaces*, 125–132.

Tony Mullen, and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP*, 412–418.

Bo Pang, and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, 271–278.

Rosalind Picard. 1997. *Affective computing*. MIT Press, Cambridge, Mass.

Dan Roth. 1998. Learning to resolve natural language ambiguities: a unified approach. In *AAAI*, 806–813.

Klaus Scherer. 2003. Vocal communication of emotion: a review of research paradigms. *Speech Commununication*, 40(1-2):227–256.

Greg Siegle. The Balanced Affective Word List *http://www.sci.sdsu.edu/CAL/wordlist/words.prn*

Oliver Steele et al. Py-WordNet *http://osteele.com/projects/pywordnet/*

Futoshi Sugimoto et al. 2004. A method to classify emotional expressions of text and synthesize speech. In *IEEE*, 611–614.

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL*, 417–424.

Jan van Santen et al. 2003. Applications of computer generated expressive speech for communication disorders. In *EUROSPEECH 2003*, 1657–1660.

Janyce Wiebe et al. 2004. Learning subjective language. *Journal of Computational Linguistics*, 30(3):277–308.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? Finding strong and weak opinion clauses. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, 761–769.

Combining Multiple Forms of Evidence While Filtering

Yi Zhang *
Information System and Technology Management
School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064, USA
yiz@soe.ucsc.edu

Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
callan@cs.cmu.edu

Abstract

This paper studies how to go beyond relevance and enable a filtering system to learn more interesting and detailed data driven user models from multiple forms of evidence. We carry out a user study using a real time web based personal news filtering system, and collect extensive multiple forms of evidence, including explicit and implicit user feedback. We explore the graphical modeling approach to combine these forms of evidence. To test whether the approach can help us understand the domain better, we use graph structure learning algorithm to derive the causal relationships between different forms of evidence. To test whether the approach can help the system improve the performance, we use the graphical inference algorithms to predict whether a user likes a document based on multiple forms of evidence. The results show that combining multiple forms of evidence using graphical models can help us better understand the filtering problem, improve filtering system performance, and handle various data missing situations naturally.

1 Introduction

An adaptive personal information filtering system is an autonomous agent that delivers information to the user in a dynamic environment over a period of time. A common filtering approach is adapting existing text classification/retrieval algorithms to classify incoming documents as either relevant or non relevant using user profiles learned from explicit user feedback on documents the user has seen. However, there are other important criteria for the user besides relevance, such as readability (Collins-Thompson and Callan, 2004), novelty (Harman, 2003), and authority (Kleinberg, 1998). Besides, much information about the user and the document can be collected by a filtering system. These suggest a way to improve the current filtering system: going beyond relevance and using multiple forms of evidence.

Unfortunately, there is no standard evaluation data set for this research, and there is not much work on finding a good theory to combine various forms of evidence. To solve the first problem, we designed a user study and collect thousands of cases with multiple forms of evidence, including the content of a document, explicit and implicit user feedback, such as a user's mouse usage, key board usage, document length, novelty, relevance, readability, authority, user profile characteristics, news source information, and whether a user likes a document or not. Solving the second problem is very challenging. A good model should have the representation power to combine multiple forms of evidence; it should be able to help us understand the relationships between various forms of evidence; it should use the evidence to improve filtering system performance; and it should handle various problems like missing data in an operational environment robustly.

On the other hand, researchers have identified three major advantages of graphical modeling approach: 1) it provides inference tools to naturally handle situations of missing data entry because of the conditional dependencies encoded in the graph structure; 2) it can learn causal relationships in the domain, thus help us to understand the problem and to predict the consequences of intervention; and 3) it can easily combine prior knowledge (such as partial information about the causal relationship) with data in this framework. This approach has been applied to model computer software users (Horvitz et al., 1998), car drivers (Pynadath and Wellman, 1995), and students (Conati et al., 1997). Motivated by the prior work, we choose to use graphical models as our solution. To understand relationships between various forms of evidence, we use the causal graph structure learning algorithms (advantage 2), together with some prior knowledge of the domain (advantage 3), to derive the causal relationships between different user feedback, actions and user context. To improve the existing filtering system, especially in the situation of missing data, we use statistical inference tools to predict how a user will like a document, using information available in different missing evidence situations (advantage 1). We also try linear regression as an alternative approach.

The following sections describe our efforts towards

*This research was done while at the Language Technologies Institute, Carnegie Mellon University.

Figure 1: The user study system structure. The structured information, such as user feedback and crawler statistics, are kept in the database. The content of each web page crawled is saved in the news repository.

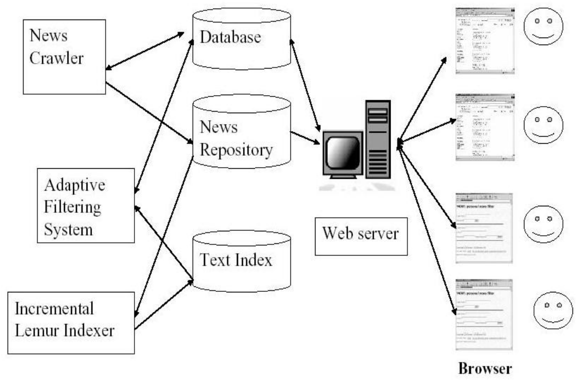
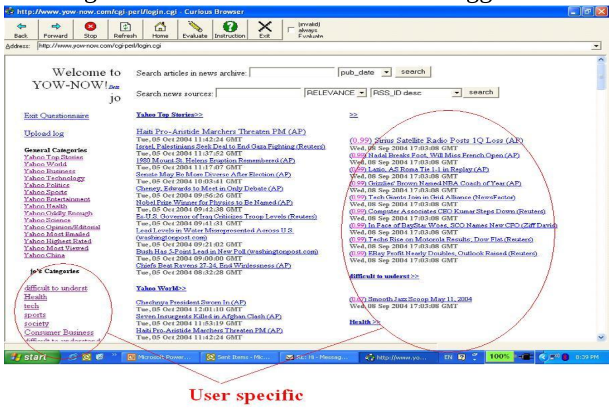


Figure 2: Web interface after a user logged in.



Address: http://www.yow-now.com/cgi-perl/login.cgi

Welcome to YOW-NOW! Beta
jo

Search articles in news archive: [ ] pub_date search

Search news sources: [ ] RELEVANCE RSS_ID desc search

Exit Questionnaire

Upload log

General Categories
Yahoo Top Stories
Yahoo World
Yahoo Business
Yahoo Technology
Yahoo Politics
Yahoo Sports
Yahoo Entertainment
Yahoo Health
Yahoo Oddly Enough
Yahoo Science
Yahoo Opinion/Editorial
Yahoo Most Emailed
Yahoo Highest Rated
Yahoo Most Viewed
Yahoo China

jo's Categories
difficult to underst
Health
tech
sports
society
Consumer Business

Yahoo Top Stories>>

Haiti Pro-Aristide Marchers Threaten PM (AP)
Tue, 05 Oct 2004 11:42:24 GMT
Israel, Palestinians Seek Deal to End Gaza Fighting (Reuters)
Tue, 05 Oct 2004 11:37:52 GMT
1980 Mount St. Helens Eruption Remembered (AP)
Tue, 05 Oct 2004 11:17:07 GMT
Senate May Be More Diverse After Election (AP)
Tue, 05 Oct 2004 10:03:41 GMT
Cheney, Edwards to Meet in Only Debate (AP)
Tue, 05 Oct 2004 09:56:26 GMT
Nobel Prize Winner for Physics to Be Named (AP)
Tue, 05 Oct 2004 09:42:38 GMT
Ex-U.S. Governor of Iraq Criticizes Troop Levels (Reuters)
Tue, 05 Oct 2004 09:41:31 GMT
Lead Levels in Water Misrepresented Across U.S. (washingtonpost.com)
Tue, 05 Oct 2004 09:21:02 GMT
Bush Has 5-Point Lead in New Poll (washingtonpost.com)
Tue, 05 Oct 2004 09:00:00 GMT
Chiefs Beat Ravens 27-24, End Winlessness (AP)
Tue, 05 Oct 2004 08:32:28 GMT

Yahoo World>>

Chechnya President Sworn In (AP)
Tue, 05 Oct 2004 12:01:10 GMT
Seven Insurgents Killed in Afghan Clash (AP)
Tue, 05 Oct 2004 11:53:19 GMT
Haiti Pro-Aristide Marchers Threaten PM (AP)
Tue, 05 Oct 2004 11:42:24 GMT

>>

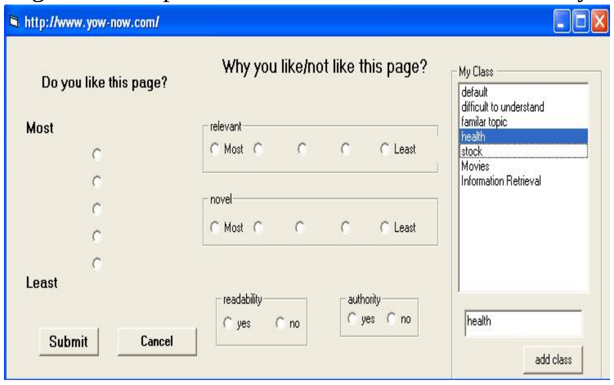
(0.99) Sirius Satellite Radio Posts 1Q Loss (AP)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) Nadal Breaks Foot, Will Miss French Open (AP)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) Lazio, AS Roma Tie 1-1 in Replay (AP)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) Grizzlies' Brown Named NBA Coach of Year (AP)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) Tech Giants Join in Grid Alliance (NewsFactor)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) Computer Associates CEO Kumar Steps Down (Reuters)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) In Face of BayStar Woes, SCO Names New CFO (Ziff Davis)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) Techs Rise on Motorola Results; Dow Flat (Reuters)
Wed, 08 Sep 2004 17:03:08 GMT
(0.99) EBay Profit Nearly Doubles, Outlook Raised (Reuters)
Wed, 08 Sep 2004 17:03:08 GMT

difficult to underst >>

(0.87) Smooth Jazz Scoop May 11, 2004
Wed, 08 Sep 2004 17:03:08 GMT

Health >>

Figure 3: Evaluation user interface. The interface for user to give their explicit feedback of the current news story.



Do you like this page?
Most
Least

Why you like/not like this page?
relevant: Most Least
novel: Most Least
readability: yes no
authority: yes no

My Class
default
difficult to understand
familar topic
health
stock
Movies
Information Retrieval

health

collecting data and customizing the graphical modeling approach to combine multiple forms of evidence for filtering. We begin with a description of the user study in Section 2, followed by some preliminary data analysis on the data collected in Section 3. Section 4 explores causal structure learning algorithm to understand the relationships between various forms of evidence from the data and Section 5 explores how to improve the system performance using multiple forms of evidence. Section 6 discusses related work and how this work differs from existing work, and Section 7 concludes.

2 User Study

No existing filtering database contains the level of detail that we needed for our study, so we developed a web based news story filtering system to collect an evaluation data set (Figure 1). This system constantly gathers and recommends information to the users. The system includes a crawler with 8000 candidate RSS news feeds (Pilgrim, 2002) to crawl every day. The Lemur indexer indexes the crawled document stream incrementally, and an adaptive filtering system recommends documents to the users using a modified logistic regression algorithm (Zhang, 2004). Users read and evaluate what the system has delivered to them. An example of the web interface after user login is in Figure 2.

More than 20 paid subjects from 19 different programs at Carnegie Mellon University, who are otherwise not affiliated with our research, participated in the study for 4 weeks. We expected to collect enough data for evaluation over this period of time. The subjects were required to read the news for about 1 hour per day and provide explicit feedback for each page they visited. [1] 28 users tried this system. However, only 21 users are official paid subjects, among which one worked only for 2 weeks and 20 worked for about 4 weeks.

2.1 Data collected

We have collected 7881 feedback entries from all 28 users, among which 7839 were from the 21 official participants. Each entry contains several different forms of evidence for a news story a user clicked.[2] Our intention to collect the evidence is not to be exhaustive, but representative. The evidence can be roughly classified into the following five categories listed in Tables 1 to 5.[3]

Explicit user feedback After finishing reading a news story, a user clicks a button on the toolbar of the browser to bring up an evaluation interface shown in Figure 3. Through this interface, the user provided the explicit feedback to tell the hidden properties about current story, including the topics the news belongs to ($classes$), how the user likes this news

[1]In the last week of the study, some subjects read 2 hours per day. They are encouraged but not required to do so.

[2]Each entry is for a <document, user class, time> tuple.

[3]The forms of evidence are listed in the first column and we will get the the other columns later in Section 3.

($user\_likes$), how relevant the news is related to the class(es) ($relevant$), how novel the news is ($novel$), whether the news matches the readability level of the user ($readable$), and whether the news is authoritative ($authoritative$). $user\_likes$, $relevant$ and $novel$ are recorded as integers ranging from 1 (least) to 5 (most). $readable$ and $authoritative$ are recorded as 0 or 1. A user has the option to provide partial instead of all explicit feedback. A user can create new classes, and choose multiple classes for one documents.

User actions The browser adapted from (Claypool et al., 2001) recorded some user actions, such as mouse activities, scroll bar activities, and keyboard activities (Table 2). TimeOnPage is the number of seconds the user spent on a page, and EventOnScroll is the number of clicks on the scroll bars. When the mouse is out of the browser window or when the browser window is not focused, the browser does not capture any activities. More details about the actions are in (Le and Waseda, 2000).

Topic information Each participant filled out an exit questionnaire and answered several topic/class[4] specific questions for each of his/her most popular 10 topics and other topics with more than 20 evaluated documents each (Table 3). The questions include how familiar the user is with the topic before the study (*topic_familiar_before*), how the user likes this topic ($topic\_like$), and how confident the user is with respect to the answers he/she provided ($topic\_confidence$). We include this information as evidence, because they may be collected when a topic is created and used by filtering systems. Whether collecting them in exit questionnaire affects the answers needs further investigation.

News Source Information For each news source (RSS feed), we collected the number of web pages that link to it (*RSS_link*), the number of pages that link to the server that provided it ($host\_link$), and the speed of the server that hosts it.

Content based evidence Three pieces of evidence are collected to represent the content of each document: the relevance score, the readability score and the number of words in the document (doc_len) (Table 5). To estimate the relevance score of a document, the system processes all the documents a user put into a class ordered by the feedback time and adaptively learns a topic specific relevance model using the relevance feedback the user provided. The relevance score of a documents is estimated using a modified logistic regression model learned from all feedback before it (Zhang, 2004). To estimate the readability score of document, the system processes all the documents in all users' classes ordered by the feedback time and adaptively learns a user independent readability model using a logistic regression algorithm.

[4]"topic" and "class" are used interchangeably in the paper.

Table 1: Basic descriptive statistics about explicit feedbacks.

Variable	Mean	variance	corr	miss
user likes	3.5	1.2	1	0.05
relevant	3.5	1.3	0.73	0.005
novel	3.6	1.33	0.70	0.008
authoritative	0.88	0.32	0.50	0.065
readable	0.90	0.30	0.54	0.012

3 Preliminary data analysis

The means and variances of all variables are in Tables 1 to 5. These basic descriptive statistics are very diverse. The values of some evidence may be missing; only the user actions and news source information were always collected. Out of the 7991 entries, only 4522 (57%) entries contain no missing value. The missing rate of each form of evidence is also reported in the tables. There are several reasons for missing data. For example, the explicit feedback is missing because users didn't always follow instructions, the relevance score is missing for the first story in a class, and the $topic\_familiar\_before$ values for many topics are missing because we only collected the topic specific answers for larger topics. We expect missing data to be common in operational environments.

The correlation coefficient between each evidence and the explicit feedback $user\_likes$ is also listed (corr). The high correlation coefficients between $user\_likes$ and other forms of explicit feedback are not very interesting because we can only get explicit feedback after a user reads the document. The correlation coefficient between relevance score and $user\_likes$ is 0.37, the highest among all forms of evidence that the system can get before delivering a document. This is not surprising since most filtering systems only consider relevance and use relevance score to make decisions.

The correlation coefficients between $user\_likes$ and the topic information (Table 3) are relatively high. This suggests collecting $topic\_familiar\_before$ or $topic\_like$ in a real filtering system, since they are informative and collecting them requires less user effort (a user only needs to provide information on the class level instead of document level). Section 5 will show how to use it with other forms of evidence in a filtering system. The correlation coefficients between the news source in-

Table 2: Basic descriptive statistics about user actions. The unit for time is second.

Variable	Mean	variance	corr
TimeOnPage	7.2×10^4	1.3×10^5	0.14
EventOnScroll	1	3.6	0.1
ClickOnWindow	0.93	2.5	0.05
TimeOnMouse	2×10^3	5.8×10^3	0.02
MSecForDownArrow	211	882	0.08
NumOfDownArrow	1.1	4.7	0.09
MSecForUpArrow	29	240	0.03
NumOfUpArrow	0.10	0.8	0.04
NumOfPageUp	0.12	0.9	$\simeq 0$
NumOfPageDown	0.14	1	$\simeq 0$
MSecForPageUp	22	202	$\simeq 0$
MSecForPageDown	28	251	$\simeq 0$

Table 3: Basic descriptive statistics about topics. Each variable ranges from 1 to 7.

variable	Mean	variance	corr	miss
topic_familiar_before	3.6	1.9	0.30	0.27
topic_like	4.9	2.0	0.30	0.27
topic_confidence	4.7	2.0	0.34	0.27

Table 4: Basic descriptive statistics about news sources.

variable	Mean	variance	corr
RSS_link	90.35	4.89	0.14
host_link	4.41×10^4	7.5×10^7	0.08
RSS_SPEED	3.92×10^5	3.7×10^9	-0.08

Table 5: Basic descriptive statistics about documents. The length of the document does not include HTML tags.

variable	mean	variance	corr	miss
doc_length	837	1.2×10^3	0.04	0.05
relevant_score	0.49	0.42	0.37	0.18
readability_score	0.52	0.16	0.25	0.11

formation and $user\_likes$ are weaker (Table 4). The correlation coefficient between $user\_likes$ and each user action (Table 2) is even lower (Table 1). Some actions, such as $TimeOnPage$, are more correlated with $user\_likes$ than other refined actions, such as $NumOfPageDown$. This finding agrees with (Claypool et al., 2001).

4 Understanding the domain using causal structure learning

Correlation analysis in Section 3 has helped us to get some initial idea about the data collected. However, in order to better understand the underlying truth of the domain, we need to go beyond correlation and uncover the causal relationships between different variables.

To do that, we first specify N nodes, one for each form of evidence to be included in the model. Then PC algorithm is used (Spirtes et al., 2000) to search the causal relationships between multiple forms of evidence from the data collected. To make the search space smaller, some prior domain knowledge, such as forbidden edges, required edges or temporal tiers, can be introduced before searching. In our experiments, we manually specified some prior knowledge based on the first authors' experience and intuition as the following 5-tier temporal tier: [5] 1) $Topic\ info\ =$ (familiar_topic_before), $RSS\ info$ =(RSS_link, host_link), document length ($doc\,len$); 2) hidden criteria, such as $relevant$, $novel$, $authoritative$, and $readable$; 3) system generated scores, such as *relevance score* and *readability score*; 4) $user\_likes$; 5) user actions, such as seconds spent on a page (TimeOnPage) or the number of clicks on the $\downarrow$ key (NumOfDownArrow). This informs the learning algorithm that $\rightarrow$ from a higher level to lower level is prohibited.

It is very encouraging to see that the structure learned automatically looks reasonable (Figure 4). According to the graph, $novel$, $relevant$, $authoritative$, $readabilty$ of a document and whether a user is familiar with the topic before using the system ($familar\_topic\_before$) are direct causes of the user's preference for a document ($user\_likes$) . How familiar with this topic a user is before participating the study ($topic\_familiar\_before$) and the number of web links to the news source (RSS_link) directly affect the user's $relevant$ and $authoritative$ feedback and $readability\ score$. $Relevant$, $authoritative$, $familiar\_topic\_before$ and $host\_link$ influence a user's actions, such as the $EventOnScroll$.

Comparing Tables 2 to 5 with Figure 4, one may ask why some variables are correlated with *user likes* although there is no direct links between them and *user likes*. For example, why the correlation between *relevance score* and *user likes* is 0.39, while there is no direct link between them. Does Figure 4 contradict Table 5? The answer is "no". In fact the indirect causal relationship between them tells us why *relevance score* and *user likes* are correlated: *relevance score* and *user likes* have a common cause *relevant*. Most of the refined actions, such as the number of pressing page up key ($NumOfPageUp$), are far away from $user\_likes$. This implies that these refined actions are not very informative if we want to use the learned model to predict whether a user likes a document or not. This finding agree with (Claypool et al., 2001) and Table 2.

The node $authoritative$ is directly linked to $readability\ score$ and $host\ link$. The link between $host\_link$ and $authoritative$ confirms the existing approaches that use the web link structure to estimate the

[5]Other priors are also possible.

Figure 4: User independent causal graphical structure learned using PC algorithm. $X \rightarrow Y$ means X is a direct cause of Y. $X - Y$ means the algorithm cannot tell if X causes Y or if Y causes X. $X \longleftrightarrow Y$ means the algorithm found some problem, which may happen due to a latent common cause of X and Y, a chance pattern in the sample, or other violations of assumptions.

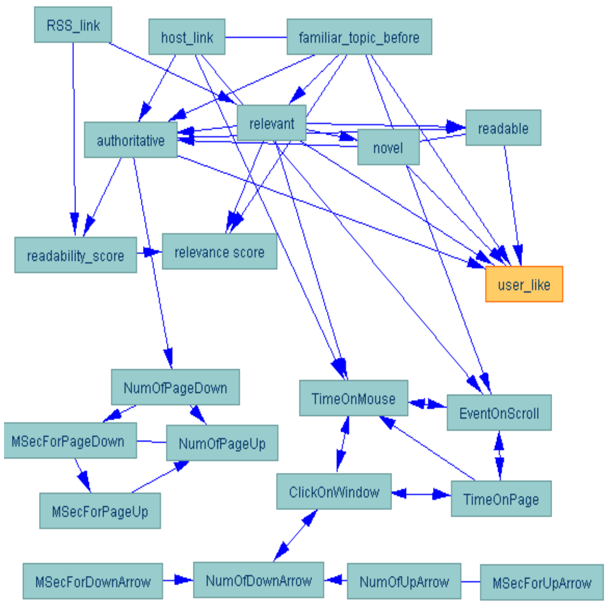


Figure 5: Structure of GM_complete.

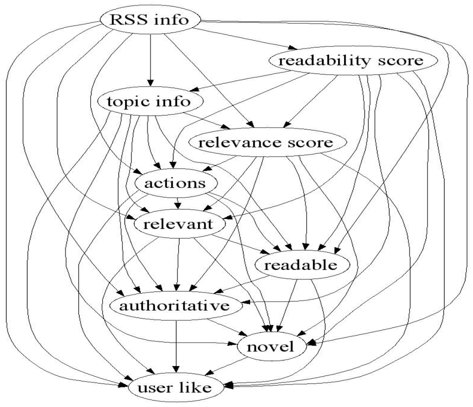
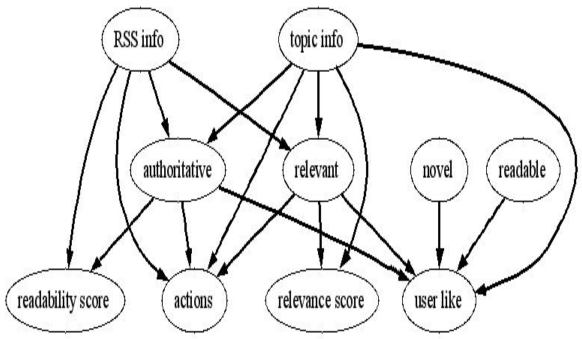


Figure 6: Structure of GM_causal.



authority of a page (Kleinberg, 1998). The links between $readability\ score$, $readable$ and $authoritative$ are very interesting. They suggest the difficulty to understand a page may make the user feel it is not authoritative. Further investigation shows that although the percentage of un-authoritative news is less than 15% in general, among the 187 news stories some users identified as "difficult" using class labels, 73% were also rated as not authoritative. Besides some successful web page authority algorithms that only use hyper links, the estimation of authority may be further improved using the content of a page.

There are links among *relevant*, *novel*, *readable* and *authoritative*. Although the algorithm failed to tell the causal direction between some pairs of variables, it suggests that the four variables influence each other. This may be an inherent property of the document; or because a user is likely to rate one aspect of the document higher than he/she should if the other aspects are good.

One may ask why the structure in Figure 4 contains no link between $readable$ and $readability\ score$, since intuitively it should exist. To answer this question, one needs to understand that the causal relationships learned automatically are what the algorithm "believes" based on the evidence of the data, the assumptions it makes, and the prior constraints we engineered. They may have errors, because the data is noisy, or the assumptions and the prior constraints may be wrong. For example, the PC algorithm do statistical test about the independence relationships among variables using the data and the final results are subject to the error of the statistical test. The PC algorithm assumes no hidden variables, however besides *relevant, novel, authoritative, and readable*, other hidden variables, such as *whether a document is up-to-date, interesting, misleading, etc.* (Schamber and Bateman, 1996), may exist and influence a user's preference for a document. Thus it is not surprising that some of the causal relationships, such as the link between $readable$ and $readability\ score$, are missed in the final graph because of the limitation of the learning algorithms. The model learned only sheds some light on the relationships between the variables instead of uncovering the whole truth. It only serves as a starting point for us. To further understand the domain, we may want to break down some variables in the current graph further and relate them to either the user or document properties. In general, causal discovery is inherently difficult and far from solved.

5 Improving system performance using inference algorithms

A primary task of a filtering system is to predict user preference ($user\_likes$) for a document so that the system can decide whether to deliver it to the user. To tell whether combining multiple forms of evidence using graphical models can improve system performance, we evaluate the proposed solution on the task of predicting $user\_likes$ while filtering.

To predict $user\_likes$, the system needs to learn a graphical model: the combination of a graph structure and a set of local conditional probability functions or potential functions. Doing inference over the causal structure learned in the previous section is difficult because of the circles and a mixture of directed and undirected links on the graph. So, we tried the following directed acyclic graphical models.

GM_complete, an almost complete Bayesian network: In this graph, we order the nodes from top to bottom, and the parents of a node are all the nodes above it, such as in Figure 5. For this structure, the order of the nodes is not very important when using Gaussian distributions.

GM_causal, a graphical model inspired by causal models: We manually modify the causal structure in Figure 4 to make it a directed acyclic graph as in Figure 6.

In the graphs, *RSS info=(RSS_link, host_link)* and *Topic info=topic_familiar_before, topic_like)* are 2 dimensional vectors representing the information about the news source and the topic in Table 4 and Table 3. $actions = (TimeOnPage, ...)$ is a 12 dimensional vector representing the user actions in Table 1. $user\_likes$ is the target variable the system wants to predict.

Before learning the parameters of the model, we need to choose a specific conditional form for the probability function associated with each node. We chose Gaussian distributions. If the parents of node X are Y, $P(X|Y) = N(m + W \times Y, \Sigma)$, where $N(\mu, \Sigma)$ is a gaussian distribution with mean μ and covariance Σ. This is a commonly used distribution for continuous valued nodes. It assumes the joint distribution of these variables is multivariate Gaussian, which may be wrong. Nevertheless, because of the mathematical convenience, the existence of efficient learning and inference algorithms for Gaussian networks, and the availability of modeling tools, we chose this distribution. Using the BNT Toolbox (Murphy, 2001), the maximum likelihood estimations of the parameters (m, W, Σ) were learned using EM algorithm and junction tree inference engine(Cowell et al., 1999) over the graphical models, with whatever information was available on the first $2/3$ of the data.

An alternative approach to combine multiple forms of evidence is linear regression. We tried two special methods to solve the missing evidence problem while using linear regression: 1) building a model that does not use the evidence that is missing for each missing situation (*LR_different*); or 2)*mean substitution*: replacing each missing value for an evidence with the average of the observed evidence (LR_mean). For K different forms of evidence, the system may need to handle 2^K different evidence missing situations. A large number of linear regression models need to be learned if we use the first approach, considering K is higher than 15 in some of our experiments. Building 2^{15} models is almost impossible for us, so a heuristic approach, which is discussed later, was used to make the experiments possible.

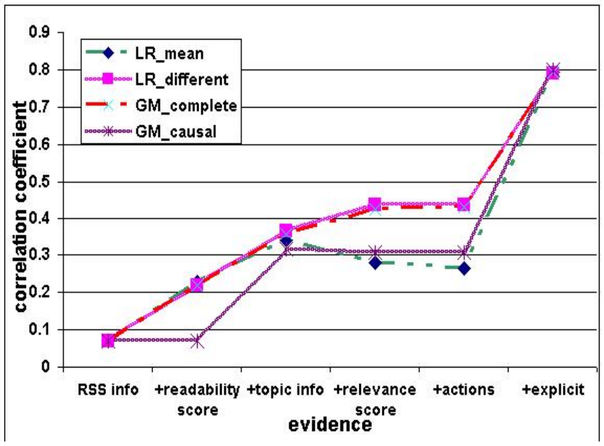


Figure 7: Comparison of the prediction power of different models using 7952 cases for evaluation. The vertical axis is the correlation coefficient between the predicted value of $user\_likes$ using the model and the true explicit feedback provided by the users. The order of different forms of evidence is set manually, based on how easy it is to collect each evidence.

Not all 7991 cases collected in the user study were used in the experiments. We conducted two sets of experiments. For the first set of experiments, we use 7952 cases for which $user\_likes$ is not missing. For the other set of runs, we use only cases without missing value. In this task, the value of each variable is continuous and normalized to variance one. Each model is learned using all information available on the first $2/3$ of the cases, and tested on the remaining $1/3$ of the cases. The correlation coefficient between the predicted value of $user\_likes$ and the true explicit $user\_likes$ feedback provided by the users is used as the evaluation measure. Our baseline is using $relevance\ score$ alone, which has a correlation coefficient of 0.367 with 95% confidence interval 0.33-0.40 on the last 1/3 of the 7952 cases.

5.1 Experimental results and discussions

Figure 7 shows the effectiveness of different models at different testing conditions as indicated by the horizontal axis. From left to right, additional sources of evidence are given when testing. At the very left of the figure (x=$RSS\ info$), a model predicts the value of $user\_likes$ only given the value of $RSS\ info$ at testing time. "+explicit" means the explicit feedback (except $user\_likes$) about the current document is given besides the value of $actions$, $relevance\ score$, $readability\ score$, $RSS\ info$, and $TopicInfo$. The graphical models and LR_mean model were trained with all evidence/features, and the learned models are independent of the testing condition. *LR_different* models were only trained with features that are also provided at testing time, so there is one model per testing condition. [6]

The results show that $GM\_complete$ performs similarly to *LR_different*. This is not surprising. Theoretically, if there is no missing entries in training data, $GM\_complete$'s estimation of the conditional distribution of $P(user\ likes|available\ evidence)$ would be the same as that of *LR_different* on a testing case with missing evidence.

Comparing the correlation coefficients under different testing conditions when using *LR_different* or $GM\_complete$, we can see that as more forms of evidence are available, the performance improves. If only the news source information of a document ($RSS\ info$) is given, all models perform poorly. The $readability\ score$ improves the system performance significantly. This is nice and interesting, because the evidence is user independent and can be estimated efficiently for each document. The performance keeps improving as $topic\ info$ and $relevance\ score$ were added. To collect them, we needs user feedback on previous documents. The performance improvement is not very obvious with $actions$ added. This means that given other evidence ($RSS\ info$, $topic\ info$, $relevance\ score$ and $readability\ score$), the system won't improve its prediction of the document much by observing these actions. However, this is only true when we use a model learned for all users and other forms of evidence are available. It does not mean the actions are useless if we learn user specific model, or if other forms of evidence (such as $relevance\ score$) are not available. All models perform very good with *explicit feedback* added. However, this is a "cheating" condition of less interest to us.

The performances of LR_mean and GM_causal do not increase monotonically as more forms of evidence are added. They perform much worse than *LR_different* and $GM\_complete$. Why does a structure that looks more causally reasonable not perform well as the simple $GM\_complete$? We may answer this question better by comparing the underlying assumptions of these algorithms. $GM\_complete$ only assumes the joint distribution of all variables is multivariate Gaussian. GM_causal makes much stronger independence assumptions by removing some links between variables. As mentioned before, the causal relationships learned automatically are not perfect, which may cause the poor performance of GM_causal. LR_mean also suffers from the strong conditional independent assumptions.

[6]However, for a specific testing condition, the training data and testing data contain cases where some evidence that is supposed to be available is missing. These cases in training data were ignored and not used to learn a $LR\_different$ model. However, ignoring such kind of cases in testing data makes comparison of different runs difficult. So we used mean substitution approach to fill the required missing features in testing data while using $LR\_different$.

Model	Cond.	corr	RLow	RUp
LR_mean	+R	0.2783	0.2426	0.3132
LR_different	+R	0.4372	0.4058	0.4677
GM_complete	+R	0.4247	0.3928	0.4555
GM_causal	+R	0.3078	0.2728	0.342
LR_mean	+A	0.2646	0.2286	0.2998
LR_different	+A	0.4375	0.406	0.4679
GM_complete	+A	0.4315	0.3999	0.4622
GM_causal	+A	0.3086	0.2736	0.3428

Table 6: A comparison of different models on all data under the $+relevance\ score$ (+R) and $+action$ (+A) conditions. Corr is the correlation coefficient between the predicted value of $user\_likes$ using the model and the true explicit feedback provided by the users. RLO and RUP are the lower and upper bounds for a 95% confidence interval for each coefficient.

Table 6 reports the performance together with the confidence intervals of all the models under the $+relevance\ score$ and $+actions$ conditions. Under both conditions, $GM\_complete$ and *LR_different* are statistically significantly better than the baseline 0.367. LR_mean and GM_causal are significantly worse. It means using multiple forms of evidence may hurt some models and benefit others. Further analysis about the $+actions$ runs shows that LR_mean gave *explicit feedback* too much weight and overlooked other less strong evidence. At testing time, it did not handle the problem of missing *explicit feedback* well and thus performed poorly. Although $GM\_complete$ also gave very high weights to *explicit feedback*, it could infer the missing values based on other available evidence at testing time, thus performed better than LR_mean. *LR_different* didn't consider *explicit feedback* for training, thus it didn't overlook other forms of evidence and suffer from the problem less. LR_mean may work reasonably if explicit variables are not included, however the large difference on how informative each evidence is will still hurt the performance of LR_mean to some extent when some strong evidence is missing. For $GM\_complete$ approach, a single model is needed to handle various evidence missing situations. If we use *LR_different* approach, several models are needed. As we mentioned before, there are 2^K

Model	Cond.	Corr	RLow	RUp
LR_mean	+R	0.13	0.08	0.18
LR_different	+R	0.41	0.37	0.45
GM_complete	+R	0.41	0.37	0.45
GM_causal	+R	0.41	0.375	0.45
LR_mean	+A	0.11	0.061	0.16
LR_different	+A	0.42	0.38	0.46
GM_complete	+A	0.42	0.38	0.46
GM_causal	+A	0.38	0.33	0.42

Table 7: The performance on 4522 no missing value cases under the $+relevance\ score$ (+R) and $+action$ (+A) conditions.

different evidence missing combinations, and 2^K linear regression models are needed in order to handle all these situations using *LR_different* approach. *LR_different* may be preferred if K is small, while graphical modeling using $GM\_complete$ may be a better approach to handle different data missing situations if K is big.

So far, all results are based on 7952 cases where some evidence may be missing. We also compared the models under different testing conditions using the 4522 cases that do not have any missing value (Table 7). GM_causal performs significantly better than before. We need to be very careful with the structures while using the graphical modeling approach, since a structure that looks more reasonable may work poorly on the inference task. However, we couldn't not draw any conclusion on whether $GM\_complete$ is better in general, because the answer may be different with different conditional probability distributions, different data sets, or a better structure learning algorithm.

6 Related Work

There has been some research on news filtering using time-coded implicit feedback (Lang, 1995; Morita and Shinoda, 1994). We noticed that an independent work uses a different graphical modeling approach, dependency network, to understand the relationships between implicit measures and explicit satisfaction while user were conducting their web searches and viewing results, and then uses decision tree to predict user satisfaction with results (Fox et al., 2005). Our work differs from the previous work in the goal of the task, the range of evidence considered, the modeling approach we took, and the findings reached.

There has been a lot of related research on using implicit feedback (Kelly and Teevan, 2003). The user actions we collected are based on (Claypool et al., 2001). There is much work about how to handle missing data. (Schafer and Graham, 2002) discussed several approaches such as case deletion, mean substitution, and recommended maximum likelihood (ML) and Bayesian multiple imputation (MI). LR_mean uses mean substitution, *LR_different* uses case deletion, and graphical models follow the ML approach.

There has been some research on criteria beyond topic relevance (Carbonell and Goldstein, 1998) (Zhang et al., 2002) (Collins-Thompson and Callan, 2004) (Kleinberg, 1998). (Schamber and Bateman, 1996) identified criteria underlying users' relevance judgements and explored how users employed the criteria in making evaluations by asking users to interpret and sort criteria independent of document manually. In the literature, the word "relevant" is used ambiguously, either as a narrow definition of "related to the matter at hand (aboutness)" or a broader definition of "having the ability to satisfy the needs of the user". When it is used by the second definition, such as in (Schamber and Bateman, 1996), researchers are usually studying what we refer to as *user likes*. In this paper, we use "relevant" as is defined in the first definition and use the phrase "user likes" for the second definition. Despite the vocabulary difference, our work is motivated by the early research. The major contributions of our work in this area are: 1) we model the *user likes* and other criteria as hidden variables; 2) we quantify the importance of various criteria based on probabilistic reasoning; and 3) we have explored the new methodology for combining these criteria with implicit and explicit user feedback.

7 CONCLUSION

We have explored how to combine multiple forms of evidence using the graphical modeling approach. This work is significant because it addresses some long-standing issues in the adaptive information filtering community: the integration of a wider range of user-specific and user-independent evidence, and handling situations like missing data that occur in operational environments.

We have analyzed the user study data using graphical models, as well as linear regression algorithms. The experimental results show that the graphical modeling approach can help us to understand the causal relationships between multiple forms of evidence in the domain and explain the real world scenario better. It can also help the filtering system to predict user preference more accurately with multiple forms of evidence compared to using a relevance model only.

As more forms of evidence are added, missing data is a common problem because of system glitches or because users will not behave as desired. A real system needs to handle missing data by either ignoring it or by estimating it based on what is known. The graphical modeling approach addresses this problem naturally. *LR_different* handles the problem by building many different models to be used at different data missing conditions. *LR_different* and $GM\_complete$ perform similarly. When the types of evidence is few, *LR_different* probably is preferable because of the simplicity. However, as more forms of evidence are added, a more powerful model, such as $GM\_complete$, may be preferred because of the computation and space efficiency.

We only collected data for documents users clicked. Further investigation is needed to look at data not clicked, which is a critical step to see whether the improvement on prediction accuracy of user preference will help the system serve the user better in a real system. This is the first step towards using graphical models to combine multiple forms of evidence while filtering. The proposed solution, especially the data analyzing methodology used in this paper, can also be used in other IR tasks besides filtering, such as context-based retrieval.

8 Acknowledgments

We thank Jaime Carbonell, Tom Minka, Stephen Robertson, Yiming Yang, Wei Xu, Peter Spirtes, Diane Kelley, Paul Ogilvie, Kevyn Collins-Thompson, Luo Si, Joemon Jose for valuable discussions about the work described in this paper.

This research was funded in part by a fellowship from IBM and a grant from National Science Foundation. Any opinions, findings, conclusions or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

References

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference*.

Mark Claypool, Phong Le, Makoto Wased, and David Brown. 2001. Implicit interest indicators. In *Intelligent User Interfaces*.

K. Collins-Thompson and J. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the HLT/NAACL 2004 Conference*.

C. Conati, A. S. Gertner, K. VanLehn, and M. J. Druzdzel. 1997. On-line student modeling for coached problem solving using Bayesian networks. In *Proceedings of the Sixth International Conference on User Modeling*, pages 231–242.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer.

Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. In *ACM Trans. Information Systems*, volume 23.

Donna Harman. 2003. Overview of the TREC 2002 novelty track. In *The Eleventh Text REtrieval Conference (TREC-11)*. NIST 500-251.

E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, July.

Diane Kelly and Jaime Teevan. 2003. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28.

J. Kleinberg. 1998. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*.

Ken Lang. 1995. Newsweeder: Learning to filter news. In *Proceedings of the Twelfth International Conference on Machine Learning*.

Phong Le and Makoto Waseda. 2000. A curious browser: Implicit ratings. http://www.cs.wpi.edu/ claypool/mqp/iii/.

Masahiro Morita and Yoichi Shinoda. 1994. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 17th ACM SIGIR conference*.

Kevyn Murphy. 2001. The Bayes net toolbox for matlab. In *Computing Science and Statistics*.

Mark Pilgrim. 2002. What is RSS. http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html.

D.V. Pynadath and W.P. Wellman. 1995. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*.

Joseph L. Schafer and John W. Graham. 2002. Missing data: Our view of the state of art. In *Psychological Methods*, volume 7, No 2.

Linda Schamber and Judy Bateman. 1996. User criteria in relevance evaluation: Toward development of a measurement scale. In *ASIS 1996 Annual Conference Proceedings*, October.

Perter Spirtes, Clark Glymour, and Richard Scheines. 2000. *Causation, Prediction, and Search*. The MIT Press.

Yi Zhang, Jamie Callan, and Tom Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference*.

Yi Zhang. 2004. Using Bayesian priors to combine classifiers for adaptive filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference*.

Handling Biographical Questions with Implicature

Donghui Feng
Information Sciences Institute
University of Southern California
Marina del Rey, CA, 90292
donghui@isi.edu

Eduard Hovy
Information Sciences Institute
University of Southern California
Marina del Rey, CA, 90292
hovy@isi.edu

Abstract

Traditional question answering systems adopt the following framework: parsing questions, searching for relevant documents, and identifying/generating answers. However, this framework does not work well for questions with hidden assumptions and implicatures. In this paper, we describe a novel idea, a cascading guidance strategy, which can not only identify potential traps in questions but further guide the answer extraction procedure by recognizing whether there are multiple answers for a question. This is the first attempt to solve implicature problem for complex QA in a cascading fashion using N-gram language models as features. We here investigate questions with implicatures related to biography facts in a web-based QA system, PowerBio. We compare the performances of Decision Tree, Naïve Bayes, SVM (Support Vector Machine), and ME (Maximum Entropy) classification methods. The integration of the cascading guidance strategy can help extract answers for questions with implicatures and produce satisfactory results in our experiments.

1 Motivation

Question Answering has emerged as a key area in natural language processing (NLP) to apply question parsing, information extraction, summarization, and language generation techniques (Clark et al., 2004; Fleischman et al., 2003; Echihabi et al., 2003; Yang et al., 2003; Hermjakob et al., 2002; Dumais et al., 2002). Traditional question answering systems adopt the framework of parsing questions, searching for relevant documents, and then pinpointing and generating answers. However, this framework includes potential dangers. For example, to answer the question "when did Beethoven get married?", a typical QA system would identify the question target to be a "Date" and would apply techniques to identify the date Beethoven got married. Since Beethoven never married, this direct approach is likely to deliver wrong answers. The trick in the question is the implicature that Beethoven got married. In the main task of QA track of TREC 2003, the performances of most systems on providing "NIL" when no answer is possible range from only 10% to 30% (Voorhees, 2003).

Just as some questions have no answer, others may have multiple answers. For instance, with "who was Ronald Reagan's wife?", a QA system may give only "Nancy Davis" as the answer. However, there is another correct answer: Jane Wyman. The problem here is the implicature in the question that Reagan only got married once.

An implicature is anything that is inferred from an utterance but that is not a condition for the truth of the utterance (Gazdar, 1979; Levinson, 1983). Implicatures in questions either waste computational effort or impair the performance of a QA system or both. Therefore, when answering questions, it is prudent to identify the questions with implicatures before processing starts.

In this paper, we describe a novel idea to solve the problem: a strategy of cascading guidance. This is the first attempt to solve implicature problem for complex QA in a cascading fashion using N-gram language models as features. The cascading guidance part is designed to be inserted immediately before the search procedure to handle questions with implicatures. It can not only first identify the potential "no answer" traps but also identify whether multiple answers for this question are likely.

To investigate the performance of the cascading guidance strategy, we here study two types of questions related to biography facts in a web-based biography QA system, PowerBio. This web-based QA system extracts biographical facts from the web obtained by querying a web search engine (Google in our case). Figure 1 provides the two types of questions we selected, which we refer to as SPOUSE_QUESTION and CHILD_QUESTION.

```
I. SPOUSE_QUESTION
   E.g. Who is <PERSON>'s wife?
        Who is <PERSON>'s husband?
        Whom did <PERSON> marry?
         …
II. CHILD_QUESTION
   E.g. Who is <PERSON>'s son?
        Who is <PERSON>'s daughter?
        Who is <PERSON>'s child?
         …
```

Figure 1. SPOUSE_QUESTION and CHILD_QUESTION

Both types of questions have implicatures to justify the use of the cascading guidance strategy. Intuitively, to answer these questions, we have two issues related to implicatures to clarify:

- Does the person have a spouse/child?
- What's the number of answers for this question? (One or many?)

We therefore create two successive classification engines in the cascading classifier.

For learning, our approach queries the search engine with every person listed in the training set, extracts related features from the documents, and trains the cascading classifiers. For application, when a new question is given, the cascading classifier is applied before activation of the search subsystem. We compare the performances of four popular classification approaches in the cascading classifier, namely Decision Tree, Naïve Bayes, SVM (Support Vector Machine), and ME (Maximum Entropy) classifications.

The paper is structured as follows: related work is discussed in Section 2. We introduce our cascading guidance technique in Section 3, including Decision Tree, Naïve Bayes and SVM (Support Vector Machine) and ME (Maximum Entropy) classifications. The experimental results are presented in Section 4. We discuss related issues and future work in Section 5.

2 Related Work

Question Answering has attracted much attention from the areas of Natural Language Processing, Information Retrieval and Data Mining (Fleischman et al., 2003; Echihabi et al., 2003; Yang et al., 2003; Hermjakob et al., 2002; Dumais et al., 2002; Hermjakob et al., 2000). It is tested in several venues, including the TREC and CLEF Question Answering tracks (Voorhees, 2003; Magnini et al., 2003). Most research efforts in the Question Answering community have focused on factoid questions and successful Question Answering systems tend to have similar underlying pipelines structures (Prager et al., 2004; Xu et al., 2003; Hovy et al., 2000; Moldovan et al., 2000).

Recently more techniques for answer extraction, answer selection, and answer validation have been proposed (Lita et al., 2004; Soricut and Brill, 2004; Clark et al., 2004).

Prager et al. (2004) proposed applying constraint satisfaction obtained by asking auxiliary questions to improve system performance. This approach requires the creation of auxiliary questions, which may be complex to automate.

Ravichandran and Hovy (2002) proposed automatically learning surface text patterns for answer extraction. However, this approach will not work if no explicit answers exist in the source. The first reason is that in that situation the anchors to learn the patterns cannot be determined. Secondly, most of the facts without explicit values are not expressed with long patterns including anchors. For example, the phrase "the childless marriage" gives enough information that a person has no child. But it is almost impossible to learn such surface text patterns following (Ravichandran and Hovy, 2002).

Reported work on question processing focuses mainly on the problems of parsing questions, determining the question target for search subsystem

(Pasca and Harabagiu, 2001; Hermjakob et al., 2000). Saquete et al. (2004) decompose complex temporal questions into simpler ones based on the temporal relationships in the question.

To date, there has been little published work on handling implicatures in questions. Just-In-Time Information Seeking Agents (JITISA) was proposed by Harabagiu (2001) to process questions in dialogue and implicatures. The agents are created based on pragmatic knowledge. Traditional answer extraction and answer fusion approaches assume the question is always correct and explicit answers do exist in the corpus. Reported work attempts to rank the candidate answer list to boost the correct one into top position. This is not enough when there may not be an answer for the question posed.

For biographical fact extraction and generation, Zhou et al. (2004) and Schiffman et al. (2001) use summarization techniques to generate human biographies. Mann and Yarowsky (2005) propose fusing the extracted information across documents to return a consensus answer. In their approach, they did not consider multiple values or no values for biography facts, although multiple facts are common for some biography attributes, such as multiple occupations, children, books, places of residence, etc. In these cases a consensus answer is not adequate.

Our work differs from theirs because we are not only working on information/answer extraction; the focus in this paper is the guidance for answer extraction of questions (or IE task for values) with implicatures. This work can be of great help for immediate biographical information extraction.

We describe details of the cascading guidance technique and investigate how it will help for question answering in Section 3.

3 Cascading Guidance Technique

We turn to the Web by querying a web search engine (Google in our case) to find evidence to create guidance for answer extraction.

3.1 Classification Procedure

The cascading classifier is applied after the name of the person and the answer types are identified. Figure 2 gives the pipeline of the classification procedure.

With the identified person name, we query the search engine (Google) to obtain the top N web pages/documents. A simple data cleaning program only keeps the content texts in the web page, which is broken up into separate sentences. Following that, topic sentences are identified with the keyword topic identification technique. For each topic we provide a list of possible related keywords and any sentences containing both the person's name (or reference) and at least one of the keywords will be selected. The required features are extracted from the topic sentences and passed to the cascading classifier as supporting evidence to generate guidance for answer extraction.

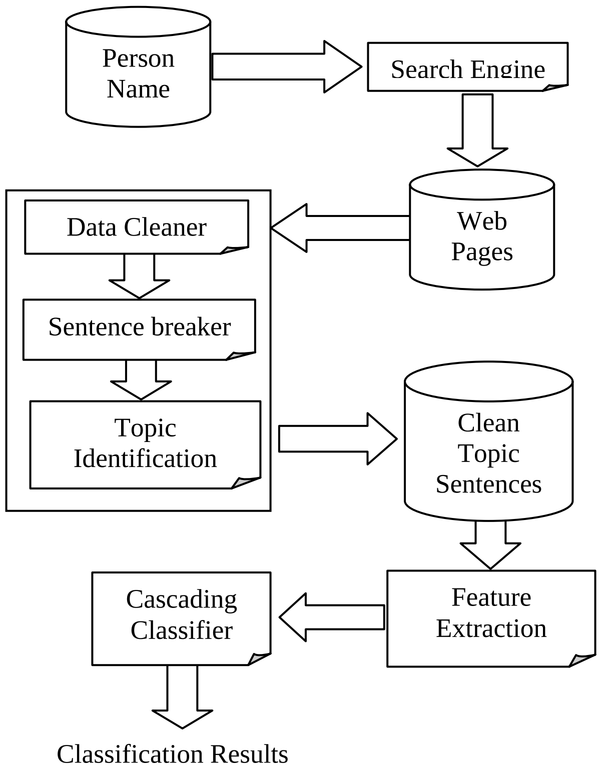


Figure 2. Procedure of Cascading Classifier

3.2 Feature Extraction

Intuitively, sentences elaborating a biographical fact in a given topic should have similar styles (short patterns) of organizing words and phrases. Here, topic means an aspect of biographical facts, e.g., marriage, children, birthplace, and so on. Inspired by this, we consider taking N-grams in sentences as our features. However, N-gram features not closely related to the topic will bring more noise into the system. Therefore, we only take the N-grams within a fixed-length window around the topic keywords for features calculation, and pass them as evidence to cascading classifier.

For N-grams, instead of using the multiplication of conditional probabilities of each word in the N-gram, we only consider the last conditional probability (see below). The reason is that the last conditional probability is a strong sign of the pattern's importance and how this sequence of words is organized. Simply multiplying all the conditional probabilities will decrease the value and require normalization. Realizing that in a set of documents the frequency of each N-gram is very important information, we combine the last conditional probability with the frequency.

The computation for each feature of unigram, bigram and trigram are defined as the following formulas:

$$f_{unigram} = p(w_i) * freq(w_i) \quad (1)$$

$$f_{bigram} = p(w_i \mid w_{i-1}) * freq(w_{i-1}, w_i) \quad (2)$$

$$f_{trigram} = p(w_i \mid w_{i-2}, w_{i-1}) * freq(w_{i-2}, w_{i-1}, w_i) \quad (3)$$

We here investigate four kinds of classifiers, namely Decision Tree, Naïve Bayes, Support Vector Machine (SVM), and Maximum Entropy (ME).

3.3 Classification Approaches

The cascading classifier is composed of two successive parts. Given the set of extracted features, the classification result could lead to different responses to the question, either answering with "no value" with strong confidence or directing the answer extraction model how many answers should be sought.

For text classification, there are several well-studied classifiers in the machine learning and natural language processing communities.

Decision Tree Classification

The Decision Tree classifier is simple and matches human intuitions perfectly while it has been proved efficient in many application systems. The basic idea is to break up the classification decision into a union of a set of simpler decisions based on N-gram features. Due to the large feature set, we use C5.0, the decision tree software package developed by RuleQuest Research (Quinlan, 1993), instead of C4.5.

Naïve Bayes Classification

The Naïve Bayes classifier utilizes Bayes' rule as follows. Supposing we have the feature set $F = \{f_1, f_2, ..., f_n\}$, the probability that person p belongs to a class c is given as:

$$c = \arg\max_{c'} P(c' \mid F) \quad (4)$$

Based on Bayes' rule, we have

$$\begin{aligned} c &= \arg\max_{c'} P(c' \mid F) \\ &= \arg\max_{c'} \frac{P(F \mid c')P(c')}{P(F)} \quad (5) \\ &= \arg\max_{c'} P(F \mid c')P(c') \end{aligned}$$

This was used for both successive classifiers of the cascading engine.

SVM Classification

SVM (Support Vector Machines) has attracted much attention since it was introduced in (Boser et al., 1992). As a special and effective approach for kernel based methods, SVM creates non-linear classifiers by applying the kernel trick to maximum-margin hyperplanes.

Suppose $p_i, i = 1, ..., n$ represent the training set of persons, and the classes for classifications are $C = \{c_1, c_2\}$ (for simplicity, we represent the classes with $C = \{-1, 1\}$). Then the classification task requires the solution of the following optimization problem (Hsu et al., 2003):

$$\begin{aligned} &\min_{\omega, b, \xi} \ \frac{1}{2}\omega^T \omega + M \sum_{i=1}^{n} \xi_i \\ &subject\ \ to \ \ c_i(\omega^T \phi(p_i) + b) \geq 1 - \xi_i \quad (6) \\ &\qquad \xi_i \geq 0 \end{aligned}$$

We use the SVM classification package LIBSVM (Chang and Lin, 2001) in our problem.

ME Classification

ME (Maximum Entropy) classification is used here to directly estimate the posterior probability for classification.

Suppose p represents the person and the classes for classifications are $C = \{c_1, c_2\}$, we have M feature functions $h_m(c, p), m = 1, ..., M$. For each feature function, we have a model parameter $\lambda_m, m = 1, ..., M$. The classification with maximum likelihood estimation can be defined as follows (Och and Ney, 2002):

$$P(c \mid p) = p_{\lambda_1^M}(c \mid p) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(c, p)]}{\sum_{c'} \exp[\sum_{m=1}^{M} \lambda_m h_m(c', p)]} \quad (7)$$

The decision rule to choose the most probable class is (Och and Ney, 2002):

$$\hat{c} = \arg\max_c \{P(c \mid p)\} = \arg\max_c \left\{ \sum_{m=1}^{M} \lambda_m h_m(c, p) \right\} \quad (8)$$

We use the published package YASMET[1] to conduct parameters training and classification. YASMET requires supervised learning for the training of maximum entropy model.

The four classification approaches are assembled in a cascading fashion. We discuss their performance next.

4 Experiments and Results

4.1 Experimental Setup

We download from infoplease.com[2] and biography.com[3] two corpora of people's biographies, which include 24,975 and 24,345 bios respectively. We scan each whole corpus and extract people having spouse information. To create the data set, we manually check and categorize each person as having multiple spouses, only one spouse, or no spouse. Similarly, we obtained another list of persons having multiple children, only one child, and no child. The sizes of data extracted are given in Table 1.

Type	Child	Spouse
No_value	25	20
One_value	35	32
Multiple_values	107	43

Table 1. Extracted experimental data

For the cascading classification, in the first step, when classifying whether a person has a spouse/child or not, we merge the last two subsets with one value and multiple values into one. Table 2 presents the data used for each level of classification.

	class	Child	Spouse
First-level Classification	No_value	25	20
	With_value	142	75
Second-level Classification	One_value	35	32
	Multiple_value	107	43

Table 2. Data set used for classification

To investigate the performances of our cascading classifiers, we divided the two sets into training set and testing set, with half of them in the training set and half in the testing set.

4.2 Empirical Results

For each situation of the two questions, when the answer type has been determined to be the child or spouse of a person, we send the person's name to Google and collect the top N documents. As described in Figure 2, topic sentences in each document are selected by keyword matching. A window with the length of *w* is applied to the sentence. All word sequences in the window are selected for feature calculation. We take all the three N-gram language models (unigram, bigram, and trigram) in the window for feature computation. Table 3 gives the sizes of the bigram feature sets for first-level classification as we take more and more documents into the system.

Top N Docs	Child	Spouse
1	3468	1958
10	27733	12325
20	46431	27331
30	61057	36637
40	76687	43771
50	87020	50868
60	96393	61632
70	108053	67712
80	118947	73306
90	130526	77370
100	139722	82339

Table 3. Sizes of feature sets

As described in Section 3, the feature values are applied in the classifiers. Tables 4 and 5 give the best performances of the 4 classifiers in the two situations when we select the top N articles using N-gram probability for feature computation.

Due to the large size of the feature set, C5.0, SVM, and ME packages will not work at some

[1] http://www.fjoch.com/YASMET.html
[2] http://www.infoplease.com/people.html
[3] http://www.biography.com/search/index.jsp

point as more documents are encountered. The Naïve Bayes classification is more scalable as we use intermediate file to store probability tables.

Precision	First-level Classification	Second-level Classification
C5.0	82.90%	65.70%
Naïve Bayes	87.80%	72.86%
SVM	84.15%	75.71%
ME	86.59%	75.71%

Table 4. Precision scores for child classification

Precision	First-level Classification	Second-level Classification
C5.0	80.90%	56.80%
Naïve Bayes	83.00%	59.46%
SVM	78.72%	54.05%
ME	78.72%	51.35%

Table 5. Precision scores for spouse classification

Feature	# of times identified (out of 75)	$p(w_i \mid w_{i-2}, w_{i-1})$
and his wife	35	0.6786
her husband ,	33	0.3082
and her husband	26	0.5476
was married to	20	0.8621
with his wife	14	0.875
her second husband	13	0.6667
her marriage to	13	0.5
ex - wife	12	0.3333
ex - husband	11	0.6667
her first husband	10	0.75
second husband ,	10	1
his first wife	8	0.3333
first husband ,	7	0.6667
second wife ,	7	0.3333
his first marriage	5	0.1667
s second wife	5	0.75

Table 6. Example trigram features for second-level classification for Spouse (one or multiple values)

The feature set has a large number of features. However, not all of them will be used for each person. We studied the number of times features are identified/used in the training and testing sets and their probabilities. Table 6 presents a list of some trigram features for second-level classification (one or multiple values) for Spouse. Obviously, indicating features have a large probability as expected. The second column gives the number of times the feature is used out of the training and testing set (75 persons in total).

Will more complex N-gram features work better?

Intuitively, being less ambiguous, more complex N-gram features carry more precise information and therefore should work better than simple ones. We studied the performances for different N-gram language model features. Below are the results of Naïve Bayes first-level classification for Child, using different N-gram features.

Top N Docs	Unigram	Bigram	Trigram
1	34.78%	54.35%	**67.39%**
10	30.48%	79.27%	**86.59%**
20	26.83%	82.93%	**85.37%**
30	24.39%	81.71%	**86.59%**

Table 7. Comparisons of classification precisions using different N-gram features for child

From Table 7, we can infer that bigram features work better than unigram features, and trigram features work better than bigrams when we select different numbers of top N documents. Trigram features actually bring enough evidence in classification. However, when we investigated 4-grams language features in the collected data, most of them are very sparse in the feature space of all the cases. Applying 4-grams or higher may not help in our task.

Will more data/documents help?

The performance of corpus-based statistical approaches usually depends on the size of corpus. A traditional view for most NLP problems is that more data will help to improve the system's performance. However, for data collected from a search engine, this may not be the case, since web data is usually ambiguous and noisy. We therefore investigate the data size's effect on system performance. Figure 3 gives the precision curves of the Naïve Bayes classifier for the first-level classification for Child.

Except for the case of top 1, where the top document alone may not contain too much useful information on selected topics, precision scores only have slight variations for increasing numbers of documents. For bigram features, over the top 50 through top 70 documents, the precision scores even get a little worse.

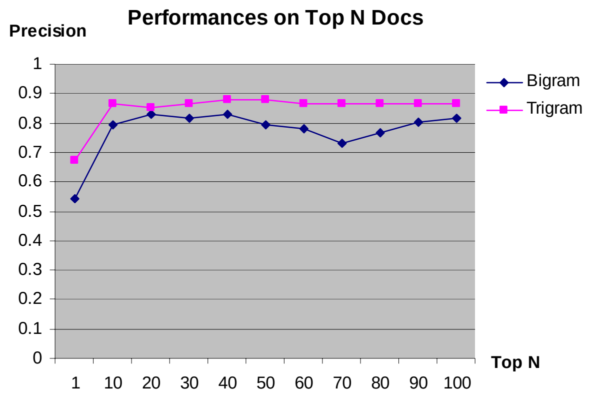


Figure 3. Performance on top N documents

4.3 Examples

Equipped with the cascading guiding strategy, we are able to handle questions containing implicatures. In our system, when we can determine the answer type is child or spouse, the cascading guiding system will help the answer extraction part to extract answers from the designated corpus. Figure 4 gives two examples of the strategy.

Q1: *Who is Sophia Smith's spouse?*
Classified: <NO_SPOUSE>
Answer: *She did not marry.*

Q2: *Who is John Ritter's child?*
Classified: <HAVING_CHILD>
Classified: <MULTIPLE_VALUES>
…

Figure 4. Classification Example for question

For the first question, the classifier recognizes there is no spouse for the target person and returns information for the answer generation. The features used here are the first-level classification result for SPOUSE_QUESTION. For the second question, the classifier recognizes the target person has a child first, followed by recognizing that the answer has multiple values. In this way, the strategy integrated to the question answering system can improve the system's performance by handling questions with implicatures.

5 Discussion and Future Work

Questions may have implicatures due to the flexibility of human language and conversation. In real question-answering systems, failure to handle them may either waste huge computation cost or impair system's performance. The traditional QA framework does not work well for questions containing implicatures. We describe a novel idea in this paper to identify potential traps in biographical questions and recognize whether there are multiple answers for a question.

Question-Answering systems, even when focused upon biographies, have to handle many facts, such as birth date, birth place, parents, training, accomplishments, etc. These values can be extracted using typical text harvesting approaches. However, when there are no values for some biographical information, the task becomes much more difficult because text seldom explicitly states a negative. For example, the following two questions require schools attended:

- *Where did <person> graduate from?*
- *What university did <person> attend?*

Our program scanned the two corpora of bios and found only 2 out 49320 bios explicitly stating that the subject never attended any school. Therefore, for some types of information, it will be much harder to identify null values through evidence from text. Some more complicated reasoning and inference may be required. Classifiers for some biographical facts may need to incorporate extra knowledge from other resources. The inherent relations between biography facts can also be used to validate each other. For example, the relations of marriage and child, birth place and childhood home, etc. may provide clues for cross-validation. We plan to investigate these problems in the future.

Acknowledgements

We wish to thank the anonymous reviewers for their helpful feedback and corrections. Also we thank Lei Ding, Feng Pan, and Deepak Ravichandran for their valuable comments on this work.

References

Boser, B.E., Guyon, I. and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. *Proceedings of the ACM COLT 1992.*

Chang, C. and Lin, C. 2001. LIBSVM -- A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Chu-Carroll, J., Czuba, K., Prager, J., and Ittycheriah, A. 2003. In question answering, two heads are better than one. *Proceedings of HLT-NAACL-2003*.

Clark, S., Steedman, M. and Curran, J.R. 2004. Object-extraction and question-parsing using CCG. *Proceedings EMNLP-2004*, pages 111-118, Barcelona, Spain.

Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. 2002. Web question answering: is more always better? *Proceedings of SIGIR-2002*.

Echihabi, A. and Marcu, D. 2003. A noisy channel approach to question answering. *Proceedings of ACL-2003*.

Fleischman, M., Hovy, E.H., and Echihabi, A. 2003. Offline strategies for online question answering: answering questions before they are asked. *Proceedings of ACL-2003.*

Gazdar, G. 1979. *Pragmatics: Implicature, presupposition, and logical form.* New York: Academic Press.

Harabagiu, S. 2001. Just-In-Time Question Answering. *Invited talk in Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium 2001.*

Hermjakob, U., Echihabi, A., and Marcu, D. 2002. Natural language based reformulation resource and web exploitation for question answering. *Proceedings of TREC-2002*.

Hermjakob, U., Hovy, E.H., and Lin, C. 2000. Knowledge-based question answering. *TREC-2000*.

Hovy, E.H., Gerber, L., Hermjakob, U., Junk, M., and Lin, C. 2000. Question answering in Webclopedia. *Proceedings of TREC-2000*.

Hovy, E.H., Hermjakob, U., Lin, C., and Ravichandran, D. 2002. Using knowledge to facilitate factoid answer pinpointing. *Proceedings of COLING-2002*.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. 2003. A Practical Guide to Support Vector Classification. Available at: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Levinson, S. 1983. *Pragmatics*. Cambridge University Press.

Lita L.V. and Carbonell, J. 2004. Instance-based question answering: a data driven approach. *Proceedings of EMNLP 2004*.

Magnini, B., Romagnoli, S., Vallin, A., Herrera, J., Peñas, A., Peinado, V., Verdejo, F., Rijke, M. 2003. The Multiple Language Question Answering Track at CLEF 2003. *CLEF 2003*: 471-486.

Mann, G. and Yarowsky, D. 2005. Multi-field information extraction and cross-document fusion. *Proceedings of ACL-2005*.

Moldovan, D., Clark, D., Harabagiu, S., and Maiorano, S. 2003. Cogex: A logic prover for question answering. *Proceedings of ACL-2003*.

Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Girju, R., Goodrum, R., and Rus, V. 2000. The structure and performance of an open-domain question answering system. *Proceedings of ACL-2000.*

Nyberg, E. et al. 2003. A multi strategy approach with dynamic planning. *Proceedings of TREC-2003*.

Och, F. J.and Ney, H. 2002. Discriminative training and maximum entropy models for statistical machine translation. *Proceedings of ACL 2002* pp. 295-302.

Pasca, M. and Harabagiu, S. 2001. High Performance Question/Answering. *Proceedings of SIGIR-2001*.

Prager, J. M., Chu-Carroll, J., and Czuba, K.W.. 2004. Question answering using constraint satisfaction. *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04).*

Quinlan, J. R. 1993. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

Ravichandran, D. and Hovy, E.H. 2002. Learning Surface Text Patterns for a Question Answering System. *Proceedings of ACL-2002.*

Saquete, E., Martínez-Barco, P., Muñoz, R., and Vicedo, J.L. 2004. Splitting complex temporal questions for question answering systems. *Proceedings of ACL'04.*

Schiffman, B., Mani, I., and Concepcion, K.J. 2001. Producing biographical summaries: combining linguistic knowledge with corpus statistics. *Proceedings of ACL/EACL-2001*.

Soricut, R. and Brill, E. 2004. Automatic question answering: beyond the factoid. *Proceedings of HLT/NAACL-2004*, Boston, MA.

Voorhees, E.M. 2003. Overview of the trec 2003 question answering track. *Proceedings of TREC-2003*.

Xu, J., Licuanan, A., Weischedel, R. 2003. TREC 2003 QA at BBN: Answering Definitional Questions. *Proceedings of TREC 2003*.

Yang, H., Chua, T.S., Wang, S., and Koh, C.K. 2003. Structured use of external knowledge for eventbased open domain question answering. *Proceedings of SIGIR-2003*.

Zhou, L., Ticrea, M., and Hovy, E.H. 2004. Multi-document biography summarization. *Proceedings of EMNLP-2004.*

The Use of Metadata, Web-derived Answer Patterns and Passage Context to Improve Reading Comprehension Performance

Yongping Du
Media Computing and Web Intelligence Laboratory
Fudan University
Shanghai, China
ypdu@fudan.edu.cn

Helen Meng
Human-Computer Communication Laboratory
The Chinese University of Hong Kong
HongKong. SAR. China
hmmeng@se.cuhk.edu.hk

Xuanjing Huang
Media Computing and Web Intelligence Laboratory
Fudan University
Shanghai, China
xjhuang@fudan.edu.cn

Lide Wu
Media Computing and Web Intelligence Laboratory
Fudan University
Shanghai, China
ldwu@fudan.edu.cn

Abstract

A reading comprehension (RC) system attempts to understand a document and returns an answer sentence when posed with a question. RC resembles the *ad hoc* question answering (QA) task that aims to extract an answer from a collection of documents when posed with a question. However, since RC focuses only on a single document, the system needs to draw upon external knowledge sources to achieve deep analysis of passage sentences for answer sentence extraction. This paper proposes an approach towards RC that attempts to utilize external knowledge to improve performance beyond the baseline set by the bag-of-words (BOW) approach. Our approach emphasizes matching of metadata (i.e. verbs, named entities and base noun phrases) in passage context utilization and answer sentence extraction. We have also devised an automatic acquisition process for Web-derived answer patterns (AP) which utilizes question-answer pairs from TREC QA, the Google search engine and the Web. This approach gave improved RC performances for both the Remedia and ChungHwa corpora, attaining *HumSent* accuracies of 42% and 69% respectively. In particular, performance analysis based on Remedia shows that relative performances of 20.7% is due to metadata matching and a further 10.9% is due to the application of Web-derived answer patterns.

1. Introduction

A reading comprehension (RC) system attempts to understand a document and returns an answer sentence when posed with a question. The RC task was first proposed by the MITRE Corporation which developed the Deep Read reading comprehension system (Hirschman et al., 1999). Deep Read was evaluated on the Remedia Corpus that contains a set of stories, each with an average of 20 sentences and five questions (of types *who, where, when, what* and *why*). The MITRE group also defined the *HumSent* scoring metric, i.e. the percentage of test questions for which the system has chosen a correct sentence as the answer. *HumSent* answers were compiled by a human annotator, who examined the stories and chose the sentence(s) that best answered the questions. It was judged that for 11% of the Remedia test questions, there is no single sentence in the story that is judged to be an appropriate answer sentence. Hence the upper bound for RC on Remedia should by 89% *HumSent* accuracy. (Hirschman et al. 1999) reported a *HumSent* accuracy of 36.6% on the Remedia test set. Subsequently, (Ng et al., 2000) used a machine learning approach of decision tree and achieved the accuracy of 39.3%. Then (Riloff and Thelen, 2000) and (Charniak et al., 2000) reported improvements to 39.7% and 41%, respectively. They made use of handcrafted heuristics such as the WHEN rule:

if contain(S, TIME), then Score(S)+=4

i.e. WHEN questions reward candidate answer sentences with four extra points if they contain a name entity TIME.

RC resembles the ad hoc question answering (QA) task in TREC.[1] The QA task finds answers to a set of questions from a *collection* of documents, while RC focuses on a single

[1] http://www.nist.gov.

document. (Light et al. 1998) conducted a detailed compared between the two tasks. They found that the answers of most questions in the TREC QA task appear more than once within the document collection. However, over 80% of the questions in the Remedia corpus correspond to answer sentences that have a single occurrence only. Therefore an RC system often has only one shot at finding the answer. The system is in dire need of extensive knowledge sources to help with deep text analysis in order to find the correct answer sentence.

Recently, many QA systems have exploited the Web as a gigantic data repository in order to help question answering (Clarke et al., 2001; Kwok et al., 2001; Radev et al., 2002). Our current work attempts to incorporate a similar idea in exploiting Web-derived knowledge to aid RC. In particular, we have devised an automatic acquisition process for Web-derived answer patterns. Additionally we propose to emphasize the importance of *metadata* matching in our approach to RC. By metadata, we are referring to automatically labeled verbs, named entities as well as base noun phrases in the passage. It is important to achieve a metadata match between the question and a candidate answer sentence before the candidate is selected as the final answer. The candidate answer sentence may be one with a high degree of word overlap with the posed question, or it may come from other sentences in the neighboring context. We apply these different techniques step by step and obtain better results than have ever previously been reported. Especially, we give experiment analysis for understanding the results.

In the rest of this paper, we will first describe three main aspects of our approach towards RC – (i) metadata matching, (ii)automatic acquisition of Web-derived answer patterns and (iii) the use of passage context. This will be followed by a description of our experiments, analysis of results and conclusions.

2. Metadata Matching

A popular approach in reading comprehension is to represent the information content of each question or passage sentence as a bag of words (BOW). This approach incorporates stopword removal and stemming. Thereafter, two words are considered a match if they share the same morphological root. Given a question, the BOW approach selects the passage sentence with the maximum number of matching words as the answer. However, the BOW approach does not capture the fact that the *informativeness* of a word about a passage sentence varies from one word to another. For example, it has been pointed out by (Charniak et al. 2000) that the verb seems to be especially important for recognizing that a passage sentence is related to a specific question. In view of this, we propose a representation for questions and answer sentences that emphasizes three types of metadata:

(i) **Main Verbs (MVerb)**, identified by the link parser (Sleator and Temperley 1993);
(ii) **Named Entities (NE)**, including names of locations (LCN), persons (PRN) and organizations (ORG), identified by a home-grown named entity identification tool; and
(iii) **Base Noun Phrases (BNP)**, identified by a home-grown base noun phrase parser respectively.

We attempt to quantify the relative importance of such metadata through corpus statistics obtained only from the training set of the Remedia corpus, which has 55 stories. The Remedia test set, which contains 60 stories, is set aside for evaluation. On average, each training story has 20 sentences and five questions. There are 274 questions in all in the entire training set. Each question corresponds to a marked answer sentence within the story text. We analyzed all the questions and divided them into three question sets (Q_SETS) based on the occurrences of MVerb, NE and BNP identified with the tools mentioned above. The following are illustrative examples of the Q_SETS as well as their sizes:

Q_SET$_{\text{Mverb}}$ (Count:169)	*Who <u>helped</u> the Pilgrims?*
Q_SET$_{\text{NE}}$ (Count:62)	*When was the first merry-go-round built in <u>the United States</u>?*
Q_SET$_{\text{BNP}}$ (Count:232)	*Where are <u>the northern lights</u>?*

Table 1. Examples and sizes of question sets (Q_SETS) with different metadata – main verb (MVerb), named entity (NE) and base noun phrase (BNP).

It may also occur that a question belongs to multiple Q_SETS. For example:

Q_SET$_{MVerb}$	*When was the first merry-go-round built in the United States?*
Q_SET$_{NE}$	*When was the first merry-go-round built in the United States?*
Q_SET$_{BNP}$	*When was the first merry-go-round built in the United States?*

Table 2. An example sentence that belongs to multiple Q_SETS.

As mentioned earlier, each question corresponds to an answer sentence, which is annotated in the story text by MITRE. Hence we can follow the Q_SETS to divide the answer sentences into three answer sets (A_SETS). Examples of A_SETS that correspond to Table 1 include:

A_SET$_{MVerb}$	*An Indian named Squanto came to help them.*
A_SET$_{NE}$	*The first merry-go-round in the United States was built in 1799.*
A_SET$_{BNP}$	*Then these specks reach the air high above the earth.*

Table 3. Examples of the answer sets (A_SETS) corresponding to the different metadata categories, namely, main verb (MVerb), named entity (NE) and base noun phrase) (BNP).

In order to quantify the relative importance of matching the three kinds of metadata between Q_SET and A_SET for reading comprehension, we compute the following relative weights based on corpus statistics:

$$Weight_{Metadata} = \frac{|S_{Metadata}|}{|A\_SET_{Metadata}|} \quad \text{.....Eqn (1)}$$

where $S_{Metadata}$ is the set of answer sentences in $|A\_SET_{Metadata}|$ that contain the metadata of its corresponding question. For example, referring to Tables 2 and 3, the question in Q_SET$_{NE}$ *"When was the first merry-go-round built in the United Sates?"* contains the named entity (underlined) which is also found in the associated answer sentence from A_SET$_{NE}$, *"The first merry-go-round in the United States was built in 1799." Hence this answer sentence belongs to the set* S$_{NE}$. Contrarily, the question in Q_SET$_{BNP}$ *"Where are the northern lights?"* contains the base noun phrase (underlined) but it is not found in the associated answer sentence from A_SET$_{BNP}$, *"Then these specks reach the air high above the earth."* Hence this answer sentence does not belong to the set S$_{BNP}$. Based on the three sets, we obtain the metadata weights:

$Weight_{MVerb}$=0.64, $Weight_{NE}$=0.38, $Weight_{BNP}$=0.21

To illustrate how these metadata weights are utilized in the RC task, consider again the question, "*Who helped the Pilgrims?*" together with three candidate answers that are "equally good" with a single word match when the BOW approach is applied. We further search for matching metadata among these candidate answers and use the metadata weights for scoring.

Question	*Who helped the Pilgrims?* MVerb identified: "help" BNP identified: "the Pilgrams"
Candidate Sentence 1	*An Indian named Squanto came to help.* Matched MVerb (underlined) Score= $Weight_{MVerb}$=0.64
Candidate Sentence 2	*By fall, the Pilgrims had enough food for the winter.* Matched BNP (underlined) Score= $Weight_{BNP}$=0.21
Candidate Sentence 3	*Then the Pilgrims and the Indians ate and played games.* Matched BNP (underlined) Score= $Weight_{BNP}$=0.21

Table 4. The use of metadata matching to extend the bag-of-words approach in reading comprehension.

3. Web-derived Answer Patterns

In addition to using metadata for RC, the proposed approach also leverages knowledge sources that are external to the core RC resources – primarily the Web and other available corpora. This section describes our approach that attempts to automatically derive answer patterns from the Web as well as score useful answer patterns to aid RC. We utilize the open domain question-answer pairs (2393 in all) from the Question Answering track of TREC (TREC8-TREC12) as a basis for automatic answer pattern acquisition.

3.1 Deriving Question Patterns

We define a set of question tags (Q_TAGS) that extend the metadata above in order to represent *question patterns*. The tags include one for main verbs (Q_MVerb), three for named entities (Q_LCN, Q_PRN and Q_ORG) and one for base noun phrases (Q_BNP). We are also careful to ensure that noun phrases tagged as named entities are not further tagged as base noun phrases.

A question pattern is expressed in terms of Q_TAGS. A question pattern can be used to represent multiple questions in the TREC QA resource. An example is shown in Table 5. Tagging the TREC QA resource provides us with a set of question patterns $\{QP_i\}$ and for each pattern, up to m_i example questions.

Question Pattern (QP_i): When do Q_PRN Q_MVerb Q_BNP?
Represented questions: Q_1: *When did Alexander Graham Bell invent the telephone?* Q_2: *When did Maytag make Magic Chef refrigerators?* Q_3: *When did Amumdsen reach the South Pole?* (m_i example questions in all)

Table 5. A question pattern and some example questions that it represents.

3.2 Deriving Answer Patterns

For each question pattern, we aim to derive answer patterns for it automatically from the Web. The set of answer patterns capture possible ways of embedding a *specific* answer in an answer sentence. We will describe the algorithm for deriving answer patterns as following and illustrate with the following question answer pair from TREC QA:

Q: *When did Alexander Graham Bell invent the telephone?*
A: *1876*

1. **Formulate the Web Query**
   The question is tagged and the Web query is formulated as "Q_TAG"+ "ANSWER", i.e.
   Question: "*When did Alexander Graham Bell invent the telephone?*"
   QP: When do Q_PRN Q_MVerb Q_BNP ?
   where Q_PRN= "*Alexander Graham Bell*", Q_MVerb= "*invent*", and Q_BNP= "*the telephone*"
   hence Web query: "*Alexander Graham Bell*"+ "*invent*" + "*the telephone*" + "*1876*"
2. **Web Search and Snippet Selection**
   The Web query is submitted to the search engine Google using the GoogleAPI and the top 100 snippets are downloaded. From each snippet, we select up to ten contiguous words to the left as well as to the right of the "ANSWER" for answer pattern extraction. The selected words must be continuous and do not cross the snippet boundary that Google denotes with '…'.
3. **Answer Pattern Selection**
   We label the terms in each selected snippet with the Q_TAGs from the question as well as the answer tag <A>. The shortest string containing all these tags (underlined below) is extracted as the answer pattern (AP). For example:

Snippet 1: *1876, Alexander Graham Bell invented the telephone in the United States…*
AP 1: <A>, Q_PRN Q_MVerb Q_BNP.
(N.B. The answer tag <A> denotes "*1876*" in this example).

Snippet 2: *…which has been invented by Alexander Graham Bell in 1876…*
AP 2: Q_MVerb by Q_PRN in <A>.

As may be seen in above, the acquisition algorithm for Web-derived answer questions calls for specific answers, such as a factoid in a word or phrase. Hence the question-answer pairs from TREC QA are suitable for use. On the other hand, Remedia is less suitable here because it contains labelled answer *sentences* instead of factoids. Inclusion of whole answer sentences in Web query formulation generally does not return the answer pattern that we seek in this work.

3.3 Scoring the Acquired Answer Patterns

The answer pattern acquisition algorithm returns *multiple* answer patterns for every question-answer pair submitted to the Web. In this subsection we present an algorithm for deriving scores for these answer patterns. The methodology is motivated by the concept of *confidence* level, similar to that used in data mining. The algorithm is as follows:

1. **Formulate the Web Query**
   For each question pattern QP_i (see Table 5) obtained previously, randomly select an example question among the m_i options that belongs to this pattern. The question is tagged and the Web query is formulated in terms of the Q_TAGs only. (Please note that the corresponding *answer is excluded* from Web query formulation here, which differs from the answer pattern acquisition algorithm). E.g.,
   Question: "*When did Alexander Graham Bell invent the telephone?*"
   Q_TAGs: Q_PRN Q_MVerb Q_BNP
   Web query: "*Alexander Graham Bell*"+ "*invent*" + "*the telephone*"
2. **Web Search and Snippet Selection**
   The Web query is submitted to the search engine

Google and the top 100 snippets are downloaded.

3. Scoring each Answer Pattern AP_{ij} relating to QP_i

Based on the question, its pattern QP_i, the answer and the retrieved snippets, totally the following counts for each answer pattern AP_{ij} relating to QP_i .

c_{ij} – # snippets matching AP_{ij} and for which the tag <A> matches the correct answer.

n_{ij} – # snippets matching AP_{ij} and for which the tag <A> matches any term

Compute the ratio $r_{ij}= c_{ij} / n_{ij}$..........Eqn(2)

Repeat steps 1-3 above for another example question randomly selected from the pool of m_i example under QP_i. We arbitrarily set the maximum number of iterations to be $k_i = \left\lceil \frac{2}{3} m_i \right\rceil$ in order to achieve decent coverage of the available examples. The confidence for AP_{ij}.is computed as

$$Confidence\ (AP_{ij}) = \frac{\sum_{i=1}^{k} r_{ij}}{k} \text{......Eqn(3)}$$

Equation (3) tries to assign high confidence values to answer patterns AP_{ij} that choose the correct answers, while other answer patterns are assigned low confidence values. E.g.:

<A>, Q_PRN Q_MVerb Q_BNP (Confidence=0.8)
Q_MVerb by Q_PRN in <A>. (Confidence=0.76)

3.4 Answer Pattern Matching in RC

The Web-derived answer patterns are used in the RC task. Based on the question and its QP, we select the related AP to match among the answer sentence candidates. The candidate that matches the highest-scoring AP will be selected. We find that this technique is very effective for RC as it can discriminate among candidate answer sentences that are rated "equally good" by the BOW or metadata matching approaches, e.g.:

Q: *When is the Chinese New Year?*
QP: When is the Q_BNP?
where Q_BNP=*Chinese New Year*
Related AP: Q_BNP is <A> (Confidence=0.82)
Candidate answer sentences 1: *you must wait a few more weeks for the Chinese New Year.*
Candidate answer sentences 2: *Chinese New Year is most often between January 20 and February 20.*

Both candidate answer sentences have the same number of matching terms – "Chinese", "New" and "Year" and the same metadata, i.e. Q_BNP=*Chinese New Year*. The term "is" is excluded by stopword removal. However the Web-derived answer pattern is able to select the second candidate as the correct answer sentence.

Hence our system gives high priority to the Web-derived AP – if a candidate answer sentence can match an answer pattern with confidence > 0.6, the candidate is taken as the final answer. No further knowledge constraints will be enforced.

4. Context Assistance

During RC, the initial application of the BOW approach focuses the system's attention on a small set of answer sentence candidates. However, it may occur the true answer sentence is not contained in this set. As was observed by (Riloff and Thelen, 2000) and (Charniak et al., 2000), the correct answer sentence often precedes/follows the sentence with the highest number of matching words. Hence both the preceding and following context sentences are searched in their work to find the answer sentence especially for *why* questions.

Our proposed approach references this idea in leveraging contextual knowledge for RC. Incorporation of contextual knowledge is very effective when used in conjunction with named entity (NE) identification. For instance, *who* questions should be answered with words tagged with Q_PRN (for persons). If the candidate sentence with the highest number of matching words does not contain the appropriate NE, it will not be selected as the answer sentence. Instead, our system searches among the *two* preceding and *two* following context sentences for the appropriate NE. Table 6 offers an illustration. Data analysis Remedia training set shows that the context window size selected is appropriate for *when*, *who* and *where* questions.

Football Catches On Fast (LATROBE, PA., September 4, 1895) - The new game of football is catching on fast, and each month new teams are being formed. *Last night was the first time that a football player was paid. The man's name is John Brallier, and he was paid $10 to take the place of someone who was hurt.…*
Question: *Who was the* first *football player to be paid?* **Sentence with maximum # matching words:** *Last night was the first time that a football player was paid.* **Correct answer sentence:** *The man's name is John Brallier, and he was paid $10 to take the place of someone who was hurt.*

Table 6. An example illustrating the use of contextual knowledge in RC.

As for *why* questions, a candidate answer sentence is selected from the context window if its first word is one of *"this"*, *"that"*, *"these"*, *"those"*, *"so"* or *"because"*. We did not utilize contextual constraints for *what* questions.

5. Experiments

RC experiments are run on the Remedia corpus as well as the ChungHwa corpus. The Remedia training set has 55 stories, each with about five questions. The Remedia test set has 60 stories and 5 questions per story. The ChungHwa corpus is derived from the book, *"English Reading Comprehension in 100 days,"* published by Chung Hwa Book Co., (H.K.) Ltd. The ChungHwa training set includes 100 English stories and each has four questions on average. The ChungHwa testing set includes 50 stories and their questions. We use *HumSent* as the prime evaluation metric for reading comprehension.

The three kinds of knowledge sources are used incrementally in our experimental setup and results are labeled as follows:

Result	Technique
Result_1	BOW
Result_2	BOW+MD
Result_3	BOW+MD+AP
Result_4	BOW+MD+AP+Context

Table 7. Experimental setup in RC evaluations. Abbrievations are: bag-of-words (BOW), metadata (MD), Web-derived answer patterns (AP), contextual knowledge (Context).

5.1 Results on Remedia

Table 8 shows the RC results for various question types in the Remedia test set.

	When	Who	What	Where	Why
Result_1	32.0%	30.0%	31.8%	29.6%	18.6%
Result_2	40.0%	28.0%	39.0%	38.0%	20.0%
Result_3	52.6%	42.8%	40.6%	38.4%	21.0%
Result_4	55.0%	48.0%	40.6%	36.4%	27.6%

Table 8. *HumSent* accuracies for the Remedia test set.

We observe that the *HumSent* accuracies vary substantially across different interrogatives. The system performs best for *when* questions and worst for *why* questions. The use of Web-derived answer patterns brought improvements to all the different interrogatives. The other knowledge sources, namely, meta data and context, bring improvements for some question types but degraded others.

Figure 1 shows the overall RC results of our system. The relative incremental gains due to the use of metadata, Web-derived answer patterns and context are 20.7%, 10.9% and 8.2% respectively. We also ran pairwise *t*-tests to test the statistical significance of these improvements and results are shown in Table 9. The improvements due to metadata matching and Web-derived answer patterns are statistically significant (p<0.05) but the improvement due to context is not.

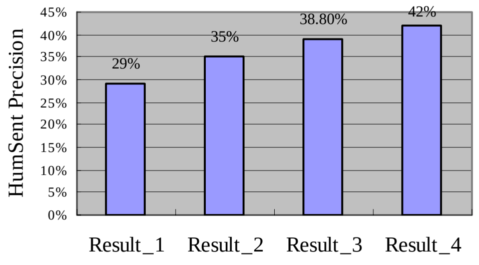


Figure 1. *HumSent* accuracies for Remedia.

Pairwise Comparison	Result_1 & Result_2	Result_2 & Result_3	Result_3 & Result_4
t-test Results	t(4)=2.207, p=0.046	t(4)=2.168, p=0.048	t(4)=1.5, p=0.104

Table 9. Tests of statistical significance in the incremental improvements over BOW among the use of metadata, Web-derived answer patterns and context.

We also compared our results across various interrogatives with those previously reported in (Riloff and Thelen, 2000). Their system is based on handcrafted rules with deterministic algorithms. The comparison (see Table 10) shows that our approach which is based on data-driven patterns and statistics can achieve comparable performance.

Question Type	Riloff &Thelen 2000	Result_4
When	55%	55.0%
Who	41%	48.0%
What	28%	40.6%
Where	47%	36.4%
Why	28%	27.6%
Overall	**40%**	**42.0%**

Table 10. Comparison of *HumSent* results with a heuristic based RC system (Riloff & Thelen 00).

5.2 Results on ChungHwa

Experimental results for the ChungHwa corpus are presented in Figure 2. The *HumSent* accuracies obtained are generally higher than those with

Remedia. We observe similar trends as before, i.e. our approach in the use of metadata, Web-derived answer patterns and context bring incremental gains to RC performance. However, the actual gain levels are much reduced.

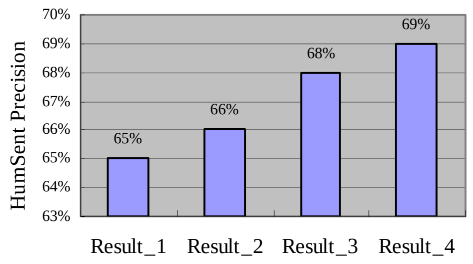


Figure 2. *HumSent* accuracies for ChungHwa.

5.3. Analyses of Results

In order to understand the underlying reason for reduced performance gains as we migrated from Remedia to Chunghwa, we analyzed the question lengths as well as the degree of word match between questions and answers among the two corpora. Figure 3 shows that the average length of questions in Chunghwa are longer than Remedia. Longer questions contain more information which is beneficial to the BOW approach in finding the correct answer.

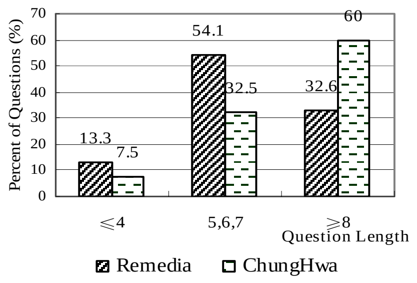


Figure 3. Distribution of question lengths among the Remedia and ChungHwa corpora.

The degree of word match between questions and answers among the two corpora is depicted in Figure 4. We observe that ChungHwa has a larger proportion of questions that have a *match- size* (i.e. number of matching words between a question and its answer) larger than 2. This presents an advantage for the BOW approach in RC. It is also observed that approximately 10% of the Remedia questions have no correct answers (i.e. match-size=-1) and about 25% have no matching words with the correct answer sentence. This explains the overall discrepancies in *HumSent* accuracies between Remedia and ChungHwa.

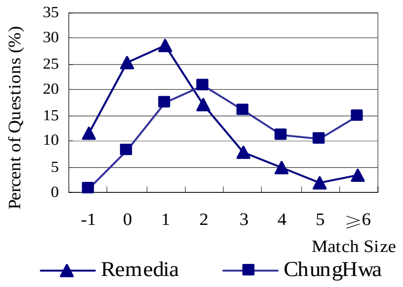


Figure 4. Distribution of match-sizes (i.e. the number of matching words between questions and their answers) in the two corpora.

While our approach has leveraged a variety of knowledge sources in RC, we still observe that our system is unable to correctly answer 58% of the questions in Remedia. An example of such elusive questions is:

Question: *When do the French celebrate their freedom?*

Answer Sentence: *To the French, July 14 has the same meaning as July 4th does to the United States.*

6. Conclusions

A reading comprehension (RC) system aims to understand a single document (i.e. story or passage) in order to be able to automatically answer questions about it. The task presents an information retrieval paradigm that differs significantly from that found in Web search engines. RC resembles the question answering (QA) task in TREC which returns an answer for a given question from a collection of documents. However, while a QA system can utilize the knowledge and information in a *collection* of documents, RC systems focuses only on a *single* document only. Consequently there is a dire need to draw upon a variety of knowledge sources to aid deep analysis of the document for answer generation. This paper presents our initial effort in designing an approach for RC that leverages a variety of knowledge sources beyond the context of the passage, in an attempt to improve RC performance beyond the baseline set by the bag-of-words (BOW) approach. The knowledge sources include the use of metadata (i.e. verbs, named entities and base noun phrases). Metadata matching is applied in our approach in answer sentence extraction as well as use of contextual sentences. We also devised an automatic acquisition algorithm for Web-derived answer patterns. The acquisition process utilizes question-answer pairs from TREC QA, the Google search engine and the Web. These answer patterns capture important structures for answer sentence extraction in RC. The use of metadata matching and Web-derived answer patterns improved reading comprehension performances for the both Remedia and ChungHwa corpora. We obtain improvements over previously reported results for Remedia, with an overall HumSet accuracy of 42%. In particular, a relative gain of 20.7% is due to metadata matching and a further 10.9% is due to application of Web-derived answer patterns.

Acknowledgement

This work is partially supported by the Direct Grant from The Chinese University of Hong Kong (CUHK) and conducted while the first author was visiting CUHK. This work is supported by Natural Science Foundation of China under Grant No.60435020.

References

Charles L.A. Clarke, Gordon V. Cormack, Thomas R. Lynam. 2001. Exploiting Redundancy in Question Answering. In Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval (SIGIR-2001, New Orleans, LA). ACM Press. New York, 358–365.

Cody C. T. Kwok, Oren Etzioni, Daniel S. Weld. 2001. Scaling Question Answering to the Web. In Proceedings of the 10th World Wide Web Conference (WWW'2001). 150-161.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. Third International Workshop on Parsing Technologies.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002). 41-47.

Dell Zhang, Wee Sun Lee. 2002. Web Based Pattern Mining and Matching Approach to Question Answering. In Proceedings of the TREC-11 Conference. 2002. NIST, Gaithersburg, MD, 505-512.

Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, Amardeep Grewal. 2002. Probabilistic Question Answering on the Web. In Proceedings of the 11th World Wide Web Conference (WWW'2002).

Ellen Riloff and Michael Thelen. 2000. A Rule-based Question Answering System for Reading Comprehension Test. ANLP/NAACL-2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems.

Eugene Charniak, Yasemin Altun, Rodrigo de Salvo Braz, Benjamin Garrett, Margaret Kosmala, Tomer Moscovich, Lixin Pang, Changhee Pyo, Ye Sun, Wei Wy, Zhongfa Yang, Shawn Zeller, and Lisa Zorn. 2000. Reading Comprehension Programs in a Statistical-Language-Processing Class. ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems.

Hwee Tou Ng, Leong Hwee Teo, Jennifer Lai Pheng Kwan. 2000. A Machine Learning Approach to Answering Questions for Reading Comprehension Tests. Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora 2000.

Lynette Hirschman, Marc Light, Eric Breck, and John Burger. 1999. Deep Read: A Reading Comprehension System. Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics.

Marc Light, Gideon S. Mann, Ellen Riloff and Eric Break. 1998. Analyses for Elucidating Current Question Answering Technology. Natural Language Engineering. Vol. 7, No. 4.

Martin M. Soubbotin, Sergei M. Soubbotin. 2002. Use of Patterns for Detection of Likely Answer Strings: A Systematic Approach. In Proceedings of the TREC-11 Conference. 2002. NIST, Gaithersburg, MD, 134-143.

Identifying Semantic Relations and Functional Properties of Human Verb Associations

Sabine Schulte im Walde and Alissa Melinger
Computational Linguistics and Psycholinguistics
Saarland University
Saarbrücken, Germany
{schulte,melinger}@coli.uni-sb.de

Abstract

This paper uses human verb associations as the basis for an investigation of verb properties, focusing on semantic verb relations and prominent nominal features. First, the lexical semantic taxonymy GermaNet is checked on the types of classic semantic relations in our data; verb-verb pairs not covered by GermaNet can help to detect missing links in the taxonomy, and provide a useful basis for defining non-classical relations. Second, a statistical grammar is used for determining the conceptual roles of the noun responses. We present prominent syntax-semantic roles and evidence for the usefulness of co-occurrence information in distributional verb descriptions.

1 Introduction

This paper presents an examination of a collection of semantic associates evoked by German verbs in a web experiment. We define semantic associates here as those concepts spontaneously called to mind by a stimulus word. In the current investigation, we assume that these evoked concepts reflect highly salient linguistic and conceptual features of the stimulus word. Given this assumption, identifying the types of information provided by speakers and distinguishing and quantifying the relationships between stimulus and response can serve a number of purposes for NLP applications.

First, the notion of semantic verb relations is crucial for many NLP tasks and applications such as verb clustering (Pereira et al., 1993; Merlo and Stevenson, 2001; Lin, 1998; Schulte im Walde, 2003), thesaurus extraction (Lin, 1999; McCarthy et al., 2003), word sense discrimination (Schütze, 1998), text indexing (Deerwester et al., 1990), and summarisation (Barzilay et al., 2002). Different applications incorporate different semantic verb relations, varying with respect to their demands. To date, limited effort has been spent on specifying the range of verb-verb relations. Morris and Hirst (2004) perform a study on lexical semantic relations which ensure text cohesion. Their relations are not specific to verb-verb pairs, but include e.g. descriptive noun-adjective pairs (such as *professors/brilliant*), or stereotypical relations (such as *homeless/drunk*). Chklovski and Pantel (2004) address the automatic acquisition of verb-verb pairs and their relations from the web. They define syntagmatic patterns to cover strength, enablement and temporal relations in addition to synonymy and antonymy, but they do not perform an exhaustive study. We suggest that an analysis of human verb-verb associations may identify the range of semantic relations which are crucial in NLP applications. We present a preparatory study where the lexical semantic taxonymy GermaNet (Kunze, 2000; Kunze, 2004) is checked on the types of classical semantic verb relations[1] in our data; verb-verb pairs not covered by GermaNet can help to detect missing links in the taxonomy, and provide an empirical basis for defining non-classical relations.

[1]We follow Morris and Hirst (2004) and refer to the paradigmatic WordNet relations as the ”classical” relations.

Second, in data-intensive lexical semantics, words are commonly modelled by distributional vectors, and the relatedness of words is measured by vector similarity. The features in the distributional descriptions can be varied in nature: words co-occurring in a document, in a context window, or with respect to a word-word relationship, such as syntactic structure, syntactic and semantic valency, etc. Most previous work on distributional similarity has either focused on a specific word-word relation (such as Pereira et al. (1993) referring to a direct object noun for describing verbs), or used any dependency relation detected by the chunker or parser (such as Lin (1999; 1998), and McCarthy et al. (2003)). Little effort has been spent on varying the (mostly nominal) types of verb features. We assume that the noun associates in our verb experiment are related to conceptual roles of the respective verbs, and investigate the linguistic functions that are realised by the response nouns with respect to the target verb, based on an empirical grammar model (Schulte im Walde, 2003). Even though the usage of the distributional features depends on the respective application, we present prominent roles and evidence for the usefulness of co-occurrence information in distributional verb descriptions.

2 Web Experiment

This section introduces our web experiment, as the data source for the explorations to follow. The web experiment asked native speakers to provide associations to German verbs.

2.1 Experiment Method

Material: 330 verbs were selected for the experiment. They were drawn from a variety of semantic classes including verbs of self-motion (e.g. *gehen* 'walk', *schwimmen* 'swim'), transfer of possession (e.g. *kaufen* 'buy', *kriegen* 'receive'), cause (e.g. *verbrennen* 'burn', *reduzieren* 'reduce'), experiencing (e.g. *hassen* 'hate', *überraschen* 'surprise'), communication (e.g. *reden* 'talk', *beneiden* 'envy'), etc. Drawing verbs from different categories was intended only to ensure that the experiment covered a wide variety of verb types; the inclusion of any verb in any particular verb class was achieved in part with reference to prior verb classification work (e.g. Levin (1993)) but also on intuitive grounds. It is not critical for the subsequent analyses. The target verbs were divided randomly into 6 separate experimental lists of 55 verbs each. The lists were balanced for class affiliation and frequency ranges (0, 100, 500, 1000, 5000), such that each list contained verbs from each grossly defined semantic class, and had equivalent overall verb frequency distributions. The frequencies of the verbs were determined by a 35 million word newspaper corpus; the verbs showed corpus frequencies between 1 and 71,604.

Procedure: The experiment was administered over the Internet. When participants loaded the experimental page, they were first asked for their biographical information, such as linguistic expertise, age and regional dialect. Next, the participant was presented with the written instructions for the experiment and an example item with potential responses. In the actual experiment, each trial consisted of a verb presented in a box at the top of the screen. All stimulus verbs were presented in the infinitive. Below the verb was a series of data input lines where participants could type their associations. They were instructed to type at most one word per line and, following German grammar, to distinguish nouns from other parts of speech with capitalisation. [2] Participants had 30 sec. per verb to type as many associations as they could. After this time limit, the program automatically advanced to the next trial.

Participants and Data: 299 native German speakers participated in the experiment, between 44 and 54 for each data set. 132 of the individuals identified themselves as having had a linguistics education and 166 rated themselves as linguistic novices. In total, we collected 81,373 associations from 16,445 trials; each trial elicited an average of 5.16 associate responses with a range of 0-16.

2.2 Data Preparation

Each completed data set contains the background information of the participant, followed by the list of target verbs. Each target verb is paired with a list of associations in the order in which the participant provided them. For the analyses to follow, we preprocessed all data sets in the following way: For each target verb, we quantified over all responses in

[2]Despite these instructions, some participants failed to use capitalisation, leading to some ambiguity.

the experiment, disregarding the participant's background and the order of the associates. Table 1 lists the 10 most frequent responses for the verb *klagen* 'complain, moan, sue'. 64% of all responses were provided more than once for a target verb, and 36% were idiosyncratic, i.e. given only once. The verb responses were not distinguished according to polysemic senses of the verbs.

klagen 'complain, moan, sue'		
Gericht	'court'	19
jammern	'moan'	18
weinen	'cry'	13
Anwalt	'lawyer'	11
Richter	'judge'	9
Klage	'complaint'	7
Leid	'suffering'	6
Trauer	'mourning'	6
Klagemauer	'Wailing Wall'	5
laut	'noisy'	5

Table 1: Association frequencies for target verb.

3 Linguistic Analyses of Experiment Data

The verb associations are investigated on three linguistic dimensions:

1. In a preparatory step, we distinguish the responses with respect to the major part-of-speech tags: nouns, verbs, adjectives, adverbs.
2. For each verb associate, we look up the semantic relation between the target and response verbs using the lexical taxonomy GermaNet.
3. For each noun associate, we investigate the kinds of linguistic functions that are realised by the noun with respect to the target verb. The analysis is based on an empirical grammar.

For expository purposes, the paper is organised into three analysis sections, with discussions following each analysis.

3.1 *Excursus:* Empirical Grammar Model

The quantitative data in the analyses to follow are derived from an empirical grammar model (Schulte im Walde, 2003): a German context-free grammar was developed with specific attention towards verb subcategorisation. The grammar was lexicalised, and the parameters of the probabilistic version were estimated in an unsupervised training procedure, using 35 million words of a large German newspaper corpus from the 1990s. The trained grammar model provides empirical frequencies for word forms, parts-of-speech tags and lemmas, and quantitative information on lexicalised rules and syntax-semantics head-head co-occurrences.

3.2 Morpho-Syntactic Analysis

The morpho-syntactic analysis is a preparatory step for the analyses to follow. Each associate of the target verb is assigned its (possibly ambiguous) part-of-speech by our empirical grammar dictionary. Originally, the dictionary distinguished approx. 50 morpho-syntactic categories, but we disregard fine-grained distinctions such as case, number and gender features and consider only the major categories verb (V), noun (N), adjective (ADJ) and adverb (ADV). Ambiguities between these categories arise e.g. in the case of nominalised verbs (such as *Rauchen* 'smoke', *Vergnügen* 'please/pleasure'), where the experiment participant could have been referring either to a verb or a noun, or in the case of past participles (such as *verschlafen*) and infinitives (such as *überlegen*), where the participant could have been referring either to a verb ('sleep' or 'think about', for the two examples respectively) or an adjective ('drowsy' or 'superior', respectively). In total, 4% of all response types are ambiguous between multiple part-of-speech tags.

Having assigned part-of-speech tags to the associates, we can distinguish and quantify the morpho-syntactic categories of the responses. In non-ambiguous situations, the unique part-of-speech receives the total target-response frequency; in ambiguous situations, the target-response frequency is split over the possible part-of-speech tags. As the result of this first analysis, we can specify the frequency distributions of the part-of-speech tags for each verb, and also in total. Table 2 presents the total numbers and specific verb examples. Participants provided noun associates in the clear majority of token instances, 62%; verbs were given in 25% of the responses, adjectives in 11%, adverbs almost never (2%).[3] The part-of-speech distribution for response words is correlated with target verb frequency. The rate of verb and adverb

[3]All of our analyses reported in this paper are based on response tokens; the type analyses show the same overall pictures.

responses is positively correlated with target verb frequency, Pearson's *r*(328)=.294, *p*<.001 for verbs and *r*(328)=.229, *p*<.001 for adverbs, while the rate of noun and adjective responses is inversely correlated with verb frequency, Pearson's *r*(328)=-.155, *p*<.005 for nouns and *r*(328)=.114, *p*<.05 for adjectives. The distribution of responses over part-of-speech also varies across verb classes. For example, aspectual verbs, such as *aufhören* 'stop', received more verb responses, *t*(12)=3.11, *p*<.01, and fewer noun responses, *t*(12)=3.84, *p*<.002, than creation verbs, such as *backen* 'bake', although the verb sets have comparable frequencies, *t*(12)=1.1, *p*<.2.

	V	N	ADJ	ADV
Total Freq	19,863	48,905	8,510	1,268
Total Prob	25%	62%	11%	2%
aufhören 'stop'	49%	39%	4%	6%
aufregen 'be upset'	22%	54%	21%	0%
backen 'bake'	7%	86%	6%	1%
bemerken 'realise'	52%	31%	12%	2%
dünken 'seem'	46%	30%	18%	1%
flüstern 'whisper'	19%	43%	37%	0%
nehmen 'take'	60%	31%	3%	2%
radeln 'bike'	8%	84%	6%	2%
schreiben 'write'	14%	81%	4%	1%

Table 2: Part-of-speech tags.

3.3 Semantic Verb Relations

For each verb associate, we look up the semantic relation between the target and response verbs using the lexical semantic taxonomy GermaNet (Kunze, 2000; Kunze, 2004), the German counterpart to WordNet (Fellbaum, 1998). The lexical database is inspired by psycholinguistic research on human lexical memory. It organises nouns, verbs, adjectives and adverbs into classes of synonyms (synsets), which are connected by lexical and conceptual relations. The GermaNet version from October 2001 contains 6,904 verbs and defines the paradigmatic semantic relations *synonymy, antonymy, hypernymy/hyponymy* as well as the non-paradigmatic relations *entailment, cause,* and *also see* between verbs or verb synsets. (*Also see* is an underspecified relation, which captures relationships other than the preceding ones. For example, *sparen* 'save' is related to *haushalten* 'budget' by *also see*.) Words with several senses are assigned to multiple synsets.

Based on the GermaNet relations, we can distinguish between the different kinds of verb associations elicited from speakers. Our analysis proceeds as follows. For each pair of target and response verbs, we look up whether any kind of semantic relation is defined between any of the synsets the verbs belong to. For example, if the target verb *rennen* 'run' is in synsets a and b, and the response verb *bewegen* 'move' is in synsets c and d, we determine whether there is any semantic relation between the synsets a and c, a and d, b and c, b and d. Two verbs belonging to the same synset are synonymous. The semantic relations are quantified by the target-response verb frequencies, e.g. if 12 participants provided the association *bewegen* for *rennen*, the hypernymy relation is quantified by the frequency 12. If the target and the response verb are both in GermaNet, but there is no relation between their synsets, then the verbs do not bear any kind of semantic relation, according to GermaNet's current status. If either of them is not in GermaNet, we cannot make any statement about the verb-verb relationship. Table 3 shows the number of semantic relations encoded in our GermaNet version, and the frequencies and probabilities of our response tokens found among them.[4] For example, there are 9,275 verb-verb instances where GermaNet defines a hypernymy-hyponymy relation between their synsets; for 2,807 of our verb-verb pairs (verb response tokens with respect to target verbs) we found a hypernymy relation among the GermaNet definitions, which accounts for 14% of all our verb responses.

The distribution of target-response relations is also correlated with target verb frequency. The proportion of associate responses captured by the respective relations of synonym, antonym and hyponym increases as a function of target verb frequency, *r*(323)=.147 for synonymy, *r*(328)=.341 for antonymy and *r*(328)=.243 for hyponymy (all *p*<.01); the proportion of hypernym relations is not correlated with verb frequency. The distribution of relations also varies by verb class. For example, aspectual target verbs like *aufhören* 'stop' received significantly more antonymic responses like *anfangen* 'begin' or *weitermachen* 'go on' than creation verbs such as *backen* 'bake', *t*(12)=3.44, *p*<.05.

[4]Note that the number of encoded relations in GermaNet differs strongly, which influences the number of verb-verb tokens that can potentially be found.

	GermaNet	Freq	Prob
Synonymy	4,633	1,194	6%
Antonymy	226	252	1%
Hypernymy	9,275	2,807	14%
Hyponymy	9,275	3,016	16%
Cause	95	49	0%
Entailment	8	0	0%
Also see	1	0	0%
No relation	-	10,509	54%
Unknown cases	-	1,726	9%

Table 3: Semantic relations.

An interesting piece of information is provided by the verb-verb pairs for which we do not find a relationship in GermaNet. The minority of such cases (9%) is due to part-of-speech confusion based on capitalisation errors by the participants, cf. footnote 2; e.g. the non-capitalised noun *wärme* ‘warmth’ was classified as a verb because it is the imperative of the verb *wärmen* ‘warm’. A remarkable number of verb-verb associations (54%) do not show any kind of semantic relation according to GermaNet despite both verbs appearing in the taxonomy. On the one hand, this is partly due to the GermaNet taxonomy not being finished yet; we find verb associations such as *weglaufen* ‘run away’ for *abhauen* ‘walk off’ (12 times), or *untersuchen* ‘examine’ for *analysieren* ‘analyse’ (8 times) where we assume (near) synonymy not yet coded in GermaNet; or *weggehen* ‘leave’ for *ankommen* ‘arrive’ (6 times), and *frieren* ‘be cold’ for *schwitzen* ‘sweat’ (2 times) where we assume antonymy not yet coded in GermaNet. For those cases, our association data provides a useful basis for detecting missing links in GermaNet, which can be used to enhance the taxonomy. However, a large proportion of the ”no relation” associations represent instances of verb-verb relations not targeted by GermaNet. For example, *adressieren* ‘address’ was associated with the temporally preceding *schreiben* ‘write’ (15 times) and the temporally following *schicken* ‘send’ (6 times); *schwitzen* ‘sweat’ was associated with a consequence *stinken* ‘stink’ (8 times) and with a cause *laufen* ‘run’ (5 times); *setzen* ‘seat’ was associated with the implication *sitzen* ‘sit’ (2 times), *erfahren* ‘get to know’ with the implication *wissen* ‘know’ (8 times). Those examples represent instantiations of non-classical verb relations and could be subsumed under *also see* relations in GermaNet, but it is obvious that one would prefer more fine-grained distinctions. We are specifically interested in those cases, because we expect that human associations cover the range of possible semantic verb relations to a large extent, and we believe that they represent an excellent basis for defining an exhaustive set, as alternative to e.g. text-based relations (Morris and Hirst, 2004). Again, the diversity of semantic verb relations is a crucial ingredient for NLP tasks such as thesaurus extraction, summarisation, and question answering.

Window Look-up We have argued above that an investigation into the types of semantic relations instantiated by verb-verb associations could be relevant in NLP. Thus, we are interested in whether paradigmatically related verb-verb pairs co-occur in texts. To evaluate this point, we perform a window look-up, in order to determine the distance between two associated verbs. We use our complete newspaper corpus, 200 million words, and check whether the response verbs occur in a window of 5/20/50 words to the left or to the right of the relevant target word. For paradigmatically related verb pairs, namely those whose relation we could determine with GermaNet (37%), we find 85/95/97% in the respective windows. For those whose relation is unspecified in GermaNet (63%), we find lower co-occurrence rates, 61/74/79%. The fact that the distances between verbs and the co-occurrance rates differ with respect to the category of semantic relation, e.g. paradigmatic or not, is useful for NLP applications such as summarisation, where both the distances between salient words and their semantic relations are crucial. More precise conditions (e.g. different window sizes, structural sentence/paragraph distinctions, quantification of window matches by their frequencies) shall be specified in future work.

3.4 Syntax-Semantic Noun Functions

In a third step, we investigate the kinds of linguistic functions that are realised by noun associates of the target verbs. Our hypothesis is that the properties of the elicited noun concepts provide insight into conceptual features for distributional verb descriptions.

The analysis utilises the empirical grammar model, cf. Section 3.1. With respect to verb subcategorisation, the grammar defines frequency distributions of verbs for 178 subcategorisation frame types, including prepositional phrase information, and frequency distributions of verbs for nominal argument fillers. For example, the verb *backen* 'bake' appeared 240 times in our training corpus. In 80 of these instances it was parsed as intransitive, and in 109 instances it was parsed as transitive subcategorising for a direct object. The most frequent nouns subcategorised for as direct objects are *Brötchen* 'rolls', *Brot* 'bread', *Kuchen* 'cake', *Plätzchen* 'cookies', *Waffel* 'waffle'.

We use the grammar information to look up the syntactic relationships which exist between a target verb and a response noun. For example, the nouns *Kuchen* 'cake', *Brot* 'bread', *Pizza* and *Mutter* 'mother' were produced in response to the target verb *backen* 'bake'. The grammar look-up tells us that *Kuchen* 'cake' and *Brot* 'bread' appear not only as the verb's direct objects (as illustrated above), but also as intransitive subjects; *Pizza* appears only as a direct object, and *Mutter* 'mother' appears only as transitive subject. The verb-noun relationships which are found in the grammar are quantified by the verb-noun association frequency, divided by the number of different relationships (to account for the ambiguity represented by multiple relationships). For example, the noun *Kuchen* was elicited 45 times in response to *bake*; the grammar contains the noun both as direct object and as intransitive subject for that verb, so both functions are assigned a frequency of 22.5. In a second variant of the analysis, we also distributed the verb-noun association frequencies over multiple relationships according to the empirical proportions of the respective relationships in the grammar, e.g. of the total association frequency of 45 for *Kuchen*, 15 would be assigned to the direct object of *backen*, and 30 to the intransitive subject if the empirical grammar evidence for the respective functions of *backen* were one vs. two thirds.

In a following step, we accumulate the association frequency proportions with respect to a specific relationship, e.g. for the direct objects of *backen* 'bake' we sum over the frequency proportions for *Kuchen, Brot, Plätzchen, Brötchen*, etc. The final result is a frequency distribution over linguistic functions for each target verb, i.e. for each verb we can determine which linguistic functions are activated by how many noun associates. For example, the most prominent functions for the inchoative-causative verb *backen* 'bake' are the transitive direct object (8%), the intransitive subject (7%) and the transitive subject (4%).

By generalising over all verbs, we discover that only 11 frame-slot combinations are activated by at least 1% of the nouns: subjects in the intransitive frame, the transitive frame (with direct/indirect object, or prepositional phrase) and the ditransitive frame; the direct object slot in the transitive, the ditransitive frame and the direct object plus PP frame; the indirect object in a transitive and ditransitive frame, and the prepositional phrase headed by *Dat:in*, dative (locative) 'in'. The frequency and probability proportions are illustrated by Table 4; the function is indicated by a slot within a frame (with the relevant slot in bold font); 'S' is a subject slot, 'AO' an accusative (direct) object, 'DO' a dative (indirect) object, and 'PP' a prepositional phrase. The activation of the functions differs only slightly with the analysis variant distributing the association frequencies with respect to grammar evidence, see above.

Interestingly, different verb classes are associated to frame slots to varying degrees. For example, verbs of creation like *backen* 'bake' elicited direct object slot fillers significantly more often than aspectual verbs like *aufhören* 'stop', *t*(12)=2.24, *p*<.05.

Function		Freq	Prob
S	**S** V	1,793	4%
	S V AO	1,065	2%
	S V DO	330	1%
	S V AO DO	344	1%
	S V PP	510	1%
AO	S V **AO**	2,298	5%
	S V **AO** DO	882	2%
	S V **AO** PP	706	1%
DO	S V **DO**	302	1%
	S V AO **DO**	597	1%
PP	S V **PP-Dat:in**	418	1%
Unknown noun		10,663	22%
Unknown function		24,536	50%

Table 4: Associates as slot fillers.

In total, only 28% of all noun associates were identified by the statistical grammar as frame-slots fillers. However, the analysis of noun functions shows that a range of linguistic functions might be considered as prominent, e.g. 11 functions are activated by more than 1% of the associates. Our hope is that these frame-role combinations are candidates for defining distributional verb descriptions. As mentioned before, most previous work on distributional similarity has focused either on a specific word-word relation (such as Pereira et al. (1993) referring to a direct object noun for describing verbs), or used any syntactic relationship detected by the chunker or parser (such as Lin (1999; 1998) and McCarthy et al. (2003)). Naturally, the contribution of distributional features depends on the distributional objects and the application, but our results suggest that it is worth determining a task-specific set of prominent features.

The majority of noun responses were not found as slot fillers. 22% of the associates are missing because they do not appear in the grammar model at all. These cases are due to (i) lemmatisation in the empirical grammar dictionary, where noun compounds such as *Autorennen* ‘car racing’ are lemmatised by their lexical heads, creating a mismatch between the full compound and its head; (ii) domain and size of training corpus, which underrepresents slang responses like *Grufties* ‘old people’, dialect expressions such as *Ausstecherle* ‘cookie-cutter’ as well as technical expressions such as *Plosiv* ‘plosive’. The remaining 50% of the nouns are represented in the grammar but do not fill subcategorised-for linguistic functions; clearly the conceptual roles of the noun associates are not restricted to the subcategorisation of the target verbs. In part what is or is not covered by the grammar model can be characterised as an argument/adjunct contrast. The grammar model distinguishes argument and adjunct functions, and only arguments are included in the verb subcategorisation and therefore found as linguistic functions. Adjuncts such as the instrument *Pinsel* ‘brush’ for *bemalen* ‘paint’ (21 times), *Pfanne* ‘pan’ for *erhitzen* ‘heat’ (2), or clause-internal information such as *Aufmerksamkeit* ‘attention’ for *bemerken* ‘notice’ (6) and *Musik* ‘music’ for *feiern* ‘celebrate’ (10) are not found. These associates fulfill scene-related roles which are not captured by subcategorisation in the grammar model. In addition, we find associates which capture clause-external scene-related information or refer to world knowledge which is not expected to be found in the context at all. For example, the association *Trockner* ‘dryer’ as the instrument for *trocknen* ‘dry’ (11 times) is not typically mentioned with the verb; similarly *Wasser* ‘water’ for *auftauen* ‘defrost’ (14), *Freude* ‘joy’ for *überraschen* ‘surprise’ (24), or *Verantwortung* ‘responsibility’ for *leiten* ‘guide’ (4) reflect world knowledge and may not be found in the immediate context of the verb.

Window Look-up Of course, the distinction between arguments, adjuncts, scene-related roles and world knowledge reflects a continuum. As a follow-up experiment, we perform a window look-up on the verb-noun pairs, in order to determine what portion of the nouns co-occur in the context of the verb and what portion is missing. This should provide us with a rough idea of the conceptual roles which are world knowledge and not found in the context. We again use our complete newspaper corpus, 200 million words, and check whether the response nouns are in a window of 5/20/50 words to the left or to the right of the relevant target verb. Naturally, most noun associates which were found as slot fillers in the functional analysis also appear in the window (since they are part of the subcategorisation): 99/99/100%. Of those cases which are not argument slot-fillers in the preceding functional analysis, we find 55/69/75% in our large corpus, i.e. more than half of the 72% missing noun tokens are in a window of 5 words from the verb, three quarters are captured by a large window of 50 words, one quarter is still missing. We conclude that, in addition to the conceptual roles referring to verb subcategorisation roles, the associations point to scene-related information and world knowledge, much of which is not explicitly mentioned in the context of the verb. With respect to a distributional feature description of verbs, we suggest that a set of prominent functions is relevant, but in addition it makes sense to include window-based nouns, which refer to scene information rather than intra-sentential syntactic functions. This is an interesting finding, since the window approach has largely been disregarded in recent years, in comparison to using syntactic functions.

4 Summary

This paper presented a study to identify, distinguish and quantify the various types of semantic associations provided by humans, and to illustrate their usage for NLP. For the approx. 20,000 verb associates, we specified classical GermaNet relations for 37% of the verb-verb pairs, and demonstrated that the co-occurrence distance between two verbs varies with respect to their semantic relation. Verb-verb pairs with no relation in GermaNet provide an empirical basis for detecting missing links in the taxonomy. Non-classical verb-verb relations such as temporal order, cause, and consequence are represented in a large proportion of the verb-verb pairs. These data represent an excellent basis for defining an exhaustive set of non-classical relations, a crucial ingredient for NLP applications.

For the approx. 50,000 noun associates, we investigated the kinds of linguistic functions that are realised by the verb-noun pairs. For 28% of the noun tokens, we found prominent frame-role combinations which speakers have in mind; our hope is that these conceptual roles represent features which contribute to distributional verb descriptions. Window-based nouns also contribute to verb descriptions by encoding scene information, rather than intra-sentential functions. This finding supports the integration of window-based approaches into function-based approaches.

Future work will establish a set of non-classical verb-verb relations, and then apply variations of verb feature descriptions in order to find dependencies between feature descriptions and verb relations. Such dependencies would improve the application of distributional verb descriptions significantly, knowing which relations are addressed by which kinds of features. In addition, we assume that the (morphological, syntactic, semantic) kinds of associates provided for a verb are indicators for its semantic class. Further investigations into the varied distributions of associate types across semantic classes will enhance the automatic acquisition of such classes. We plan to investigate this issue in more detail.

References

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2002. Inferring Strategies for Sentence Ordering in Multidocument News Summarization. *Journal of Artificial Intelligence Research*.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*.

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

Claudia Kunze. 2000. Extension and Use of GermaNet, a Lexical-Semantic Database. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*.

Claudia Kunze. 2004. Semantische Relationstypen in GermaNet. In Stefan Langer and Daniel Schnorbusch, editors, *Semantik im Lexikon*. Gunter Narr Verlag.

Beth Levin. 1993. *English Verb Classes and Alternations*. The University of Chicago Press.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 17th International Conference on Computational Linguistics*.

Dekang Lin. 1999. Automatic Identification of Non-compositional Phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a Continuum of Compositionality in Phrasal Verbs. In *Proceedings of the ACL-SIGLEX Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*.

Paola Merlo and Suzanne Stevenson. 2001. Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics*.

Jane Morris and Graeme Hirst. 2004. Non-Classical Lexical Semantic Relations. In *Proceedings of the HLT Workshop on Computational Lexical Semantics*.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional Clustering of English Words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*.

Sabine Schulte im Walde. 2003. *Experiments on the Automatic Induction of German Semantic Verb Classes*. Ph.D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*.

Accurate Function Parsing

Paola Merlo
Department of Linguistics
University of Geneva
1204 Geneva
Switzerland
merlo@lettres.unige.ch

Gabriele Musillo
Depts of Linguistics and Computer Science
University of Geneva
1204 Geneva
Switzerland
musillo4@etu.unige.ch

Abstract

In this paper, we extend an existing parser to produce richer output annotated with function labels. We obtain state-of-the-art results both in function labelling and in parsing, by automatically relabelling the Penn Treebank trees. In particular, we obtain the best published results on semantic function labels. This suggests that current statistical parsing methods are sufficiently general to produce accurate shallow semantic annotation.

1 Introduction

With recent advances in speech recognition, parsing, and information extraction, some domain-specific interactive systems are now of practical use for tasks such as question-answering, flight booking, or restaurant reservation (Stallard, 2000). One of the challenges ahead lies in moving from hand-crafted programs of limited scope to robust systems independent of a given domain. While this ambitious goal will remain in the future for some time to come, recent efforts to develop language processing systems producing richer semantic outputs will likely be the cornerstone of many successful developments in natural language understanding.

In this paper, we present a parser that outputs labels indicating the syntactic or semantic function of a constituent in the tree, such as NP-SBJ or PP-TMP shown in bold face in the tree in Figure 1. These labels indicate that the NP is the subject of the sentence and that the PP conveys temporal information. (Labels in parentheses will be explained later in the paper.) Output annotated with such informative labels underlies all domain-independent question answering (Jijkoun et al., 2004) or shallow semantic interpretation systems (Collins and Miller, 1998; Ge and Mooney, 2005). We test the hypothesis that a current statistical parser can output such richer information without any degradation of the parser's accuracy on the original parsing task. Briefly, our method consists in augmenting a state-of-the-art statistical parser (Henderson, 2003), whose architecture and properties make it particularly adaptive to new tasks. We achieve state-of-the-art results both for parsing and function labelling.

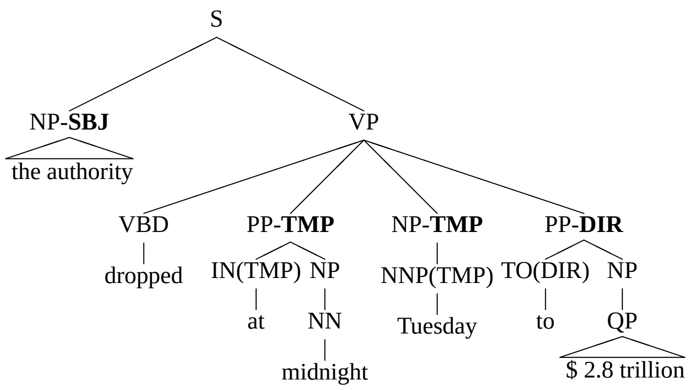


Figure 1: A sample syntactic structure with function labels.

Statistical parsers trained on the Penn Treebank (PTB) (Marcus et al., 1993) produce trees annotated with bare phrase structure labels (Collins, 1999; Charniak, 2000). The trees of the Penn Treebank, however, are also decorated with function labels. Figure 1 shows the simplified tree representation with function labels for a sample sentence from the Penn Treebank corpus (section 00) *The Government's borrowing authority dropped at midnight Tuesday to 2.8 trillion from 2.87 trillion.* Table 1 illustrates the complete list of function labels in the Penn Treebank. Unlike phrase structure labels, func-

Syntactic Labels		**Semantic Labels**	
DTV	dative	ADV	adverbial
LGS	logical subject	BNF	benefactive
PRD	predicate	DIR	direction
PUT	compl of *put*	EXT	extent
SBJ	surface subject	LOC	locative
VOC	vocative	MNR	manner
Miscellaneous Labels		NOM	nominal
CLF	*it*-cleft	PRP	purpose or reason
HLN	headline	TMP	temporal
TTL	title	**Topic Labels**	
CLR	closely related	TPC	topicalized

Table 1: Complete set of function labels in the Penn Treebank.

tion labels are context-dependent and encode a shallow level of phrasal and lexical semantics, as observed first in (Blaheta and Charniak, 2000).[1] To a large extent, they overlap with semantic role labels as defined in PropBank (Palmer et al., 2005).

Current statistical parsers do not use this richer information because performance of the parser usually decreases considerably, since a more complex task is being solved. (Klein and Manning, 2003), for instance report a reduction in parsing accuracy of an unlexicalised PCFG from 77.8% to 72.9% if using function labels in training. (Blaheta, 2004) also reports a decrease in performance when attempting to integrate his function labelling system with a full parser. Conversely, researchers interested in producing richer semantic outputs have concentrated on two-stage systems, where the semantic labelling task is performed on the output of a parser, in a pipeline architecture divided in several stages (Gildea and Jurafsky, 2002). See also the common task of (CoNLL, 2004 2005; Senseval, 2004).

Our approach maintains state-of-the-art results in parsing, while also reaching state-of-the-art results in function labelling, by suitably extending a Simple Synchrony Network (SSN) parser (Henderson, 2003) into a single integrated system. This is an interesting result, as a task combining function labelling and parsing is more complex than simple parsing. While the function of a constituent and its structural position are often correlated, they sometimes diverge. For example, some nominal temporal modifiers occupy an object position without being objects, like *Tuesday* in the tree above. Moreover, given current limited availability of annotated tree banks, this more complex task will have to be solved with the same overall amount of data, aggravating the difficulty of estimating the model's parameters due to sparse data.

[1](Blaheta and Charniak, 2000) talk of function *tags*. We will instead use the term function *label*, to indicate function identifiers, as they can decorate any node in the tree. We keep the word *tag* to indicate only those labels that decorate preterminal nodes in a tree – part-of-speech tags– as is standard use.

2 Method

Successfully addressing function parsing requires accurate parsing models and training data. Understanding the causes and the relevance of the observed results requires appropriate evaluation measures. In this section, we describe the methodology that will be used to assess our main hypothesis.

2.1 The Basic Parsing Architecture

Our main hypothesis says that function labels can be successfully and automatically recovered while parsing, without affecting negatively the performance of the parser. It is possible that attempting to solve the function labelling and the parsing problem at the same time would require modifying existing parsing models, since their underlying independence assumptions might no longer hold. Moreover, many more parameters are to be estimated. It is therefore important to choose a statistical parser that can model our augmented labelling problem. We use a family of statistical parsers, the Simple Synchrony Network (SSN) parsers (Henderson, 2003), which crucially do not make any explicit independence assumptions, and learn to smooth across rare feature combinations. They are therefore likely to adapt without much modification to the current problem. This architecture has shown state-of-the-art performance and is very adaptive to properties of the input.

The architecture of an SSN parser comprises two components, one which estimates the parameters of a stochastic model for syntactic trees, and one which searches for the most probable syntactic tree given the parameter estimates. As with many other statistical parsers (Collins, 1999; Charniak, 2000), the model of parsing is history-based. Its events are derivation moves. The set of well-formed sequences of derivation moves in this parser is defined by a Predictive LR pushdown automaton (Nederhof, 1994), which implements a form of left-corner parsing strategy.[2]

The probability of a phrase-structure tree is equated to the probability of a finite (but unbounded) sequence of derivation moves. To bound the number of parameters, standard history-based models partition the set of prefixes of well-formed sequences of transitions into equivalence classes. While such a partition makes the problem of searching for the most probable parse polynomial, it introduces hard independence assumptions: a derivation move only depends on the equivalence class to which its history belongs. SSN parsers, on the other hand, do not state any explicit independence assumptions: they induce a finite history representation of an unbounded sequence of moves, so that the representation of a move $i-1$ is included in the inputs to the representation of the next move i, as explained in more detail in (Henderson, 2003). SSN parsers only impose soft inductive biases to capture relevant properties of the derivation, thereby exhibiting adaptivity to the input. The art of designing SSN parsers consists in selecting and introducing such biases. To this end, it is sufficient to specify features that extract some information relevant to the next derivation move from previous ones, or some set of nodes that are structurally local to the node on top of the stack. These features and these nodes are input to the computation of a hidden history representation of the sequence of previous derivation moves. Given the hidden representation of a derivation, a log-linear distribution over possible next moves is computed. Thus, the set D of structurally local nodes and the set f of predefined features determine the inductive bias of an SSN system. Unless stated otherwise, for each of the experiments reported here, the set D that is input to the computation of the history representation of the derivation moves $d_1, \ldots, d_{i-1}$ includes the following nodes: top_i, the node on top of the pushdown stack before the ith move; the left-corner ancestor of top_i; the leftmost child of top_i; and the most recent child of top_i, if any. The set of features f includes the last move in the derivation, the label or tag of top_i, the tag-word pair of the most recently shifted word, the leftmost tag-word pair that top_i dominates.

[2]The derivation moves include: projecting a constituent with a specified label, attaching one constituent to another, and shifting a tag-word pair onto the pushdown stack.

2.2 The Set of Function Labels

The bracketting guidelines for the Penn Treebank II list 20 function labels, shown in Table 1 (Bies et al., 1995). Based on their description in the Penn Treebank guidelines, we partition the set of function labels into four classes, as indicated in the table. Following (Blaheta and Charniak, 2000), we refer to the first class as syntactic function labels, and to the second class as semantic function labels. In the rest of the paper, we will ignore the other two classes, for they do not intersect with PropBank labels, and they do not form natural classes. Like previous work (Blaheta and Charniak, 2000), we complete the sets of syntactic and semantic labels by labelling constituents that do not bear any function label with a NULL label.[3]

2.3 Evaluation

To evaluate the performance of our function parsing experiments, we will use several measures. First of all, we apply the standard Parseval measures of labelled recall and precision to a parser whose training data contains the Penn Treebank function labels, to assess how well we solve the standard phrase structure parsing problem. We call these figures FLABEL-less figures in the tables below and we will call the task the (simple) parsing task in the rest of the paper. Second, we measure the accuracy of this parser with an extension of the Parseval measures of labelled precision and recall applied to the set of complex labels —the phrase structure non-terminals augmented with function labels— to evaluate how well the parser solves this complex parsing problem. These are the FLABEL figures in the tables below. We call this task the function parsing task. Finally, we also assess function labelling performance on its own. Note that the maximal precision or recall score of function labelling is strictly smaller than one-hundred percent if the precision or the recall of

[3]Strictly speaking, this label corresponds to two NULL labels: SYN-NULL and SEM-NULL. A node bearing the SYN-NULL label is a node that does not bear any other syntactic label. Analogously, the SEM-NULL label completes the set of semantic labels. Note that both the SYN-NULL label and the SEM-NULL are necessary, since both a syntactic and a semantic label can label a given constituent.

		ASSIGNED LABELS										
		ADV	BNF	DIR	EXT	LOC	MNR	NOM	PRP	TMP	SEM-NULL	SUM
ACTUAL LABELS	ADV	143	0	0	0	0	0	0	1	3	11	158
	BNF	0	0	0	0	0	0	0	0	0	1	1
	DIR	0	0	39	0	3	4	0	0	1	51	98
	EXT	0	0	0	37	0	0	0	0	0	17	54
	LOC	0	0	1	0	345	3	0	0	15	148	512
	MNR	0	0	0	0	3	35	0	0	16	40	94
	NOM	2	0	0	0	0	0	88	0	0	4	94
	PRP	0	0	0	0	0	0	0	54	1	33	88
	TMP	18	0	1	0	24	11	0	1	479	105	639
	SEM-NULL	12	0	13	5	81	28	12	24	97	20292	20564
	SUM	175	0	54	42	456	81	100	80	612	20702	22302

Table 2: Confusion matrix for simple baseline model, tested on the validation set (section 24 of PTB).

the parser is less than one-hundred percent. Following (Blaheta and Charniak, 2000), incorrectly parsed constituents will be ignored (roughly 11% of the total) in the evaluation of the precision and recall of the function labels, but not in the evaluation of the parser. Of the correctly parsed constituents, some bear function labels, but the overwhelming majority do not bear any label, or rather, in our notation, they bear a NULL label. To avoid calculating excessively optimistic scores, constituents bearing the NULL label are not taken into consideration for computing overall recall and precision figures. NULL-labelled constituents are only needed to calculate the precision and recall of other function labels. (In other words, NULL-labelled constituents never contribute to the numerators of our calculations.) For example, consider the confusion matrix M in Table 2, which reports scores for the semantic labels recovered by the baseline model described below. Precision is computed as $\frac{\sum_{i\in\{\text{ADV}\cdots\text{TMP}\}} M[i,i]}{\sum_{j\in\{\text{ADV}\cdots\text{TMP}\}} M[\text{SUM},j]}$. Recall is computed analogously. Notice that $M[n,n]$, that is the $[\text{SEM-NULL},\text{SEM-NULL}]$ cell in the matrix, is never taken into account.

3 Learning Function Labels

In order to assess the complexity of the task of predicting function labels while parsing, we run first the SSN on the function parsing task, without modifications to the parser. The confusion matrix for semantic function labels of this simple baseline model is illustrated in Table 2.

It is apparent that the baseline model's largest cause of error is confusion between the labels and the NULL label. These misclassfications affect recall in particular. Consider, for example, the MNR label, where 40 out of 94 occurrences are not given a function label. We add two augmentations to the parser to alleviate this problem.

The simple baseline parser treats NULL labels like other labels, and it does not distinguish subtypes of NULL labels. Our first augmentation of the parser is designed to discriminate among constituents with these NULL labels. We hypothesize that the label NULL (ie. SYN-NULL and SEM-NULL) is a mixture of types, which will be more accurately learnt separately. If the label NULL is learnt more precisely, the recall of the other labels will increase. The NULL label in the training set was automatically split into the mutually exclusive labels CLR, OBJ and OTHER. Constituents were assigned the OBJ label according to the conditions stated in (Collins, 1999).[4]

Another striking property of the simple baseline function parser is that the SSN tends to project NULL labels more than any other label. Since SSNs decide the label of a non-terminal at projection, this behaviour indicates that the parser does not have enough information at this point in the parse to project the correct function label. We hypothesize that finer-grained labelling will improve parsing performance. This observation is consistent with results reported in (Klein and Manning, 2003), who showed that part-of-speech tags occurring in the Treebank are not fine-grained enough to discriminate between

[4]Roughly, an OBJ non-terminal is an NP, SBAR or S whose parent is an S, VP or SBAR. Any such non-terminal must not bear either syntactic or semantic function labels, or the CLR label. In addition, the first child following the head of a PP is marked with the OBJ label.

preterminals. For example, the tag TO labels both the preposition *to* and the infinitival marker. Extending (Klein and Manning, 2003)'s technique to function labelling, we split some part-of-speech tags into tags marked with semantic function labels. More precisely, we concentrate on the function labels DIR, LOC, MNR, PRP or TMP, which appear to cause the most trouble to the parser, as illustrated in Table 2.

The label attached to a non-terminal was propagated down to the pre-terminal tag of its head. The labels in parentheses in Figure 1 illustrate the effect of this lowering of the labels. The goal of this tag-splitting is to indicate more clearly to the parser what kind of label to project on reading a word-tag pair in the input. To this end, re-labelling is applied only if the non-terminal dominates the pre-terminal immediately. This constraint guarantees that only those non-terminals that are actual projections of the pre-terminal are affected by this tag-splitting method. Linguistically, we are trying to capture the notion of maximal projection. [5] This augmented model has a total of 188 non-terminals to represent labels of constituents, instead of the 33 of the original SSN parser. As a result of lowering the five function labels, 83 new part-of-speech tags were introduced to partition the original tagset of the Treebank. There are 819 tag-word pairs in this model, while the original SSN parser has a vocabulary size of 508 tag-word pairs. These augmented tags as well as the 155 new non-terminals are included in the set f of features input to parsing decisions as described in section 2.1.

SSN parsers do not tag their input sentences. To provide the augmented model with tagged input sentences, we trained an SVM tagger whose features and parameters are described in detail in (Gimenez and Marquez, 2004). Trained on section 2-21, the tagger reaches a performance of 95.8% on the test set (section 23) of the PTB using our new tag set.

4 Experiments

In this section, we report the results of the experiments testing hypotheses concerning our function parser. All SSN function parsers were trained on sections 2-21 from the Penn Treebank, validated on section 24, and tested on section 23. All models are trained on parse trees whose labels include function labels. Both results taking function labels into account (FLABEL) and results not taking them into account (FLABEL-less) are reported. All our models, as well as the parser described in (Henderson, 2003), are run only once. [6] These results are reported in Table 3.

	FLABEL			FLABEL-less		
	F	R	P	F	R	P
	Validation Set					
Base	83.4	82.8	83.9	87.7	87.1	88.2
Aug	84.6	84.0	85.2	88.1	87.5	88.7
	Test Set					
Aug	86.1	85.8	86.5	88.9	88.6	89.3
H03				88.6	88.3	88.9

Table 3: Percentage F-measure (F), recall (R), and precision (P) of the SSN baseline (Base) and augmented (Aug) parsers. H03 indicates the model illustrated in (Henderson, 2003).

Our hypothesis states that we can perform function labelling and parsing at the same time, without loss in parsing performance. For this to be an interesting statement, we need to show that function labelling is not a straightforward extension of simple parsing. If simple parsing could be easily applied to function parsing, we should not have a degradation of an SSN parser model evaluated on the complex labels, compared to the same SSN parser evaluated only on phrase structure labels. As the results on the validation set indicate, our baseline model with function labels (FLABEL) is indeed lower than the performance of the parser when function labels are not taken into account (FLABEL-less), indicating that the function parsing task is more difficult than the simple parsing task.

Since the function parsing problem is more difficult than simple parsing, it is then interesting to observe that performance of the augmented parser increases significantly (FLABEL column) ($p < .001$) without losing accuracy on the parsing task

[5] This condition was relaxed in a few cases to capture constructs such as coordinated PPs (e.g. [PP-LOC[PP[IN*at*] ...][CC*and*][PP[IN*in*] ...] ...] or infinitival clauses (e.g. [S-PRP[VP[TO*to*][VP[VB...] ...] ...]).

[6] This explains the little difference in performance between our results for H03 and those cited in (Henderson, 2003), where the best of three runs on the validation set is chosen.

(FLABEL-less column), compared to the initial parsing performance (as indicated by the performance of H03). Notice that, numerically, we do in fact a little better than H03, but this difference is not significant.[7]

Beside confirming that learning function labels does not increase parsing errors, we can also confirm that the nature of the errors remains the same. A separate comparison of labelled and unlabelled scores of our complex function parser indicates that unlabelled results are roughly 1% better than labelled results (F measure 89.8% on the validation set). The original SSN parser exhibits the same differential. This shows that, like other simple parsers, the function parser makes mostly node attachment mistakes rather than labelling mistakes.

A separate experiment only discriminating NULL labels indicates that this modification is indeed useful, but not as much as introducing new tags, on which we concentrate to explain the results. There is converging evidence indicating that the improvement in performance is due to having introduced new tag-word pairs, and not simply new words. First of all, of the 311 new tag-word pairs only 122 introduce truly new words. The remaining pairs are constituted by words that were already in the original vocabulary and have been retagged, or by tags associated to unknown words.

Second, this interpretation of the results is confirmed by comparing different ways of enlarging the vocabulary size input to the SSN. (Henderson, 2003) tested the effect of larger input vocabulary on SSN performance by changing the frequency cut-off that selects the input tag-word pairs. A frequency cut-off of 200 yields a vocabulary of 508 pairs, while a cut-off of 20 yields 4242 pairs, 3734 of which comprise new words. This difference in input size does not give rise to an appreciable difference in performance. On the contrary, we observe that introducing 122 new words and 83 new tags improves results considerably. This leads us to conclude that the performance of the augmented model is not simply due to a larger vocabulary.

We think that our tag-word pairs are effective because they are selected by a linguistically meaningful criterion and are more informative exemplars for the parser. Instead, simply decreasing the frequency cut-off adds mostly types of words for which the parser already possesses enough evidence (in general, nouns). Our method of lowering function labels acts as a finer-grained classification that partitions different kinds of complements based on their lexical semantic characteristics, yielding classes that are relevant to constituent structure. For instance, it is well known that lexical semantic properties of arguments of verbs are related to the verb's argument structure, and consequently to the parse tree that the verb occurs in. Partitioning a verb's complements into function classes could influence attachment decisions beneficially. We also think that the parser we use is particularly able to take advantage of these subclasses. One of the main properties of SSN parsers is that they do not need large vocabularies, because the SSN is good at generalising item-specific properties into an internal hidden representation of word classes.

	Syntactic Labels			Semantic Labels		
	F	R	P	F	R	P
			Validation Set			
Base	95.3	93.9	96.7	73.1	70.2	76.3
Aug	95.7	95.0	96.5	80.1	77.0	83.5
			Test Set			
Aug	96.4	95.3	97.4	86.3	82.4	90.5
BC00	95.7	95.8	95.5	79.0	77.6	80.4
B04 FT	95.9	95.3	96.4	83.4	80.3	86.7
B04 KP	98.7	98.4	99.0	78.0	73.2	83.5

Table 4: Percentage F-measure (F), recall (R), and precision (P) function labelling, separated for syntactic and semantic labels, for our models and Blaheta and Charniak's (BC00) and Blaheta's models (B04 FT, B04 KP). The feature trees (FT) and kernel perceptrons (KP) are optimised separately for the two different sets of labels.

Finally, to provide a meaningful and complete evaluation of the parser, it is necessary to examine the level of performance on the function labels for those constituents that are correctly parsed according to the usual Parseval measure, i.e. for those constituents for which the phrase structure labels and the string covered by the label have been correctly

[7]Significance was measured with the randomized significance test described in (Yeh, 2000).

	Baseline		Augmented	
	P	R	P	R
ADV	81.7	90.5	87.9	81.0
DIR	72.2	39.8	77.0	48.5
EXT	88.1	68.5	86.8	63.5
LOC	75.7	67.4	78.9	74.6
MNR	43.2	37.2	74.0	55.7
NOM	88.0	93.6	88.7	93.1
PRP	67.5	61.4	74.4	65.9
TMP	78.3	75.0	89.6	83.7

Table 5: Percentage F-measure (F), recall (R), and precision (P) function labelling, separated for individual semantic labels, for validation set.

recovered. Clearly, our parsing results would be uninteresting if our recall on function labels were very low. In that case, we would have failed to learn the function parsing task, and that would trivially yield a good performance on the simple parsing task. Table 4 reports the aggregated numbers for the baseline and the augmented model, while Table 5 reports separate figures for each semantic function label. These tables show that we also perform well on the labelling task alone. [8] Comparison to other researchers (last three lines of Table 4) shows that we achieve state-of-the-art results with a single integrated model that is jointly optimised for all the different types of function labels and for parsing, while previous attempts are optimised separately for the two different sets of labels. In particular, our method performs better on semantic labels.

5 Related Work

As far as we are aware, there is no directly comparable work, as nobody has so far attempted to fully merge function labelling or semantic role labelling into parsing. We will therefore discuss separately those pieces of work that have made limited use of function labels for parsing (Klein and Manning, 2003), and those that have concentrated on recovering function labels as a separate task (Blaheta and Charniak, 2000; Blaheta, 2004). We cannot discuss here the large recent literature on semantic role labelling for reasons of space, apart from work that also recovers function labels (Jijkoun and de Rijke, 2004) and work that trains a parser on Propbank labels as the first stage of a semantic role labelling pipeline (Yi and Palmer, 2005).

(Klein and Manning, 2003) and, to a much more limited extent, (Collins, 1999) are the only researchers we are aware of who used function labels for parsing. In both cases, the aim was actually to improve parser performance, consequently only few carefully chosen labels were used. (Klein and Manning, 2003) suggest the technique of tag splitting for the constituent bearing the label TMP. They also speculate that locative labels could be fruitfully percolated down the tree onto the preterminals. Results in Table 5 indicate more precisely that lowering locative labels does indeed bring about some improvement, but not as much as the MNR and TMP labels.

In work that predates the availability of Framenet and Propbank, (Blaheta and Charniak, 2000) define the task of function labelling for the first time and highlight its relevance for NLP. Their method is in two-steps. First, they parse the Penn Treebank using a state-of-the-art parser (Charniak, 2000). Then, they assign function labels using features from the local context, mostly limited to two levels up the tree and only one next label. (Blaheta, 2004) extends on this method by developing specialised feature sets for the different subproblems of function labelling and slightly improves the results, as reported in Table 4. (Jijkoun and de Rijke, 2004) approach the problem of enriching the output of a parser in several steps. The first step applies memory-based learning to the output of a parser mapped to dependency structures. This step learns function labels. Only aggregated results for all function labels, and not only for syntactic or semantic labels, are provided. Although they cannot be compared directly to our results, it is interesting to notice that they are slightly better in F-measure than Blaheta's (F=88.5%). (Yi and Palmer, 2005) share the motivation of our work, although they apply it to a different task. Like the current work, they observe that the distributions of semantic labels could potentially interact with the distributions of syntactic labels and redefine the boundaries of constituents, thus yielding trees that reflect generalisations over both these sources of information.

[8]See also (Musillo and Merlo, 2005) for more detail and comparisons on the labelling task.

6 Conclusions

In this paper we have presented a technique to extend an existing parser to produce richer output, annotated with function labels. We show that both state-of-the-art results in function labelling and in parsing can be achieved. Application of these results are many-fold, such as information extraction or question answering where shallow semantic annotation is necessary. The technique illustrated in this paper is of wide applicability to all other semantic annotation schemes available today, such as Propbank and Framenet, and can be easily extended. Work to extend this technique to Propbank annotation is underway. Since function labels describe dependence relations between the predicative head and its complements, whether they be arguments or adjuncts, this paper suggests that a left-corner parser and its probabilistic model, which are defined entirely on configurational criteria, can be used to produce a dependency output. Consequences of this observation will be explored in future work.

Acknowledgments

We thank the Swiss National Science Foundation for its support of this research under grant number 105286. We thank James Henderson for allowing us to use his parser and for numerous helfpul discussions.

References

Ann Bies, M. Ferguson, K.Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style. Technical report, University of Pennsylvania.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Procs of NAACL'00*, pages 234–240, Seattle, Washington.

Don Blaheta. 2004. *Function Tagging*. Ph.D. thesis, Department of Computer Science, Brown University.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Procs of NAACL'00*, pages 132–139, Seattle, Washington.

Michael Collins and Scott Miller. 1998. Semantic tagging using a probabilistic context-free grammar. In *Procs of the Sixth Workshop on Very Large Corpora*, pages 38–48, Montreal, CA.

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

CoNLL. 2004, 2005. Conference on computational natural language learning (conll-2004/05).

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Procs of CONLL-05*, Ann Arbor, Michigan.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Jesus Gimenez and Lluis Marquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Procs of LREC'04*, Lisbon, Portugal.

Jamie Henderson. 2003. Inducing history representations for broad-coverage statistical parsing. In *Procs of NAACL-HLT'03*, pages 103–110, Edmonton, Canada.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Procs of ACL'04*, pages 311–318, Barcelona,Spain.

Valentin Jijkoun, Maarten de Rijke, and Jori Mur. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Procs of COLING-2004*, Geneva, Switzerland.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Procs of ACL'03*, pages 423–430, Sapporo, Japan.

Mitch Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Gabriele Musillo and Paola Merlo. 2005. Assigning function labels to unparsed text. In *Procs of RANLP'05*, Korovets, Bulgaria.

Mark Jan Nederhof. 1994. *Linguistic Parsing and Program Transformations*. Ph.D. thesis, Department of Computer Science, University of Nijmegen.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–105.

Senseval. 2004. Third international workshop on the evaluation of systems for the semantic analysis of text (acl 2004). http://www.senseval.org/senseval3.

David Stallard. 2000. Talk'n'travel: A conversational system for air travel planning. In *Procs of ANLP'00*, pages 68–75, Seattle, Washington.

Alexander Yeh. 2000. More accurate tests for the statistical significance of the result differences. In *Procs of COLING 2000*, pages 947–953, Saarbrucken, Germany.

Szu-ting Yi and Martha Palmer. 2005. The integration of semantic parsing and semantic role labelling. In *Procs of CoNLL'05*, Ann Arbor, Michigan.

Recognising Textual Entailment with Logical Inference

Johan Bos
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
jbos@inf.ed.ac.uk

Katja Markert
School of Computing
University of Leeds
Woodhouse Lane
Leeds, LS2 9JT
markert@comp.leeds.ac.uk

Abstract

We use logical inference techniques for recognising textual entailment. As the performance of theorem proving turns out to be highly dependent on not readily available background knowledge, we incorporate model building, a technique borrowed from automated reasoning, and show that it is a useful robust method to approximate entailment. Finally, we use machine learning to combine these deep semantic analysis techniques with simple shallow word overlap; the resulting hybrid model achieves high accuracy on the RTE testset, given the state of the art. Our results also show that the different techniques that we employ perform very differently on some of the subsets of the RTE corpus and as a result, it is useful to use the nature of the dataset as a feature.

1 Introduction

Recognising textual entailment (RTE) is the task to find out whether some text T entails a hypothesis H. This task has recently been the focus of a challenge organised by the PASCAL network in 2004/5.[1] In Example 1550 H follows from T whereas this is not the case in Example 731.

[1]All examples are from the corpus released as part of the RTE challenge. It is downloadable from `http://www.pascal-network.org/Challenges/RTE/`. The example numbers have also been kept. Each example is marked for entailment as TRUE if H follows from T and FALSE otherwise. The dataset is described in Section 4.1.

Example: 1550 (TRUE)

T: In 1998, the General Assembly of the Nippon Sei Ko Kai (Anglican Church in Japan) voted to accept female priests.

H: The Anglican church in Japan approved the ordination of women.

Example: 731 (FALSE)

T: The city Tenochtitlan grew rapidly and was the center of the Aztec's great empire.

H: Tenochtitlan quickly spread over the island, marshes, and swamps.

The recognition of textual entailment is without doubt one of the ultimate challenges for any NLP system: if it is able to do so with reasonable accuracy, it is clearly an indication that it has some thorough understanding of how language works. Indeed, recognising entailment bears similarities to Turing's famous test to assess whether machines can think, as access to different sources of knowledge and the ability to draw inferences seem to be among the primary ingredients for an intelligent system. Moreover, many NLP tasks have strong links to entailment: in summarisation, a summary should be entailed by the text; paraphrases can be seen as mutual entailment between T and H; in IE, the extracted information should also be entailed by the text.

In this paper, we discuss two methods for recognising textual entailment: a shallow method relying mainly on word overlap (Section 2), and deep semantic analysis, using state-of-the-art off-the-shelf inference tools, namely a theorem prover and a model builder (Section 3). These tools rely on Discourse Representation Structures for T and H as well as lexical and world knowledge. To our knowledge, few approaches to entailment currently use theorem provers and none incorporate model building (see

Section 5 for a discussion of related work).

Both methods are domain-independent to increase transferrability and have not been tailored to any particular test suite. In Section 4 we test their accuracy and robustness on the RTE datasets as one of the few currently available datasets for textual inference. We also combine the two methods in a hybrid approach using machine learning. We discuss particularly the following questions:

- Can the methods presented improve significantly over the baseline and what are the performance differences between them? Does the hybrid system using both shallow and deep semantic analysis improve over the individual use of these methods?
- How far does deep semantic analysis suffer from a lack of lexical and world knowledge and how can we perform logical inference in the face of potentially large knowledge gaps?
- How does the design of the test suite affect performance? Are there subsets of the test suite that are more suited to any particular textual entailment recognition method?

2 Shallow Semantic Features

We use several shallow surface features to model the text, hypothesis and their relation to each other.

Most importantly, we expect some dependency between surface string similarity of text and hypothesis and the existence of entailment. Our string similarity measure uses only a form of extended word overlap between text and hypothesis, taking into account equality of words, synonymy and morphological derivations. WordNet (Fellbaum, 1998) is used as the knowledge source for synonymy and derivations. The exact procedure is as follows:

Both text and hypothesis are tokenised and lemmatised. A lemma l_1 in the hypothesis is said to be *related* to a lemma l_2 in the text iff l_1 and l_2 are equal, belong to the same WordNet synset (e.g., "murder" and "slay"), are related via WordNet derivations (e.g. "murder" and "murderer") or are related via a combination of synonymy and derivations (e.g. "murder" via "murderer" to "liquidator"). No word sense disambiguation is performed and *all* synsets for a particular lemma are considered.

In addition, each lemma in the hypothesis is assigned its inverse document frequency, accessing the Web as corpus via the GoogleAPI, as its weight. This standard procedure allows us to assign more importance to less frequent words.

The overlap measure `wnoverlap` between text and hypothesis is initialised as zero. Should a lemma in the hypothesis be related to a lemma in the text, its weight is added to `wnoverlap`, otherwise it is ignored. In the end `wnoverlap` is normalised by dividing it by the sum of all weights of the lemmas in the hypothesis. This ensures that `wnoverlap` is always a real number between 0 and 1 and also ensures independence of the length of the hypothesis.

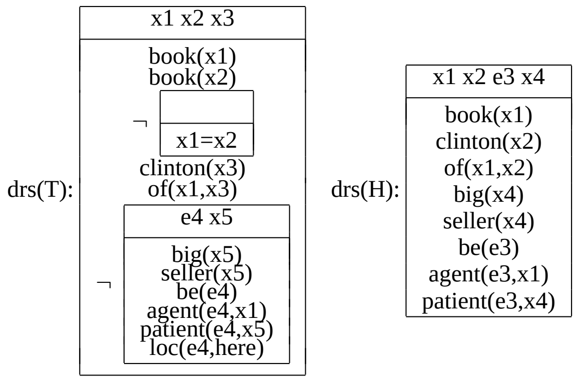
Apart from `wnoverlap` we take into account length (as measured by number of lemmas) of text and hypothesis, because in most of the observed cases for true entailments the hypothesis is shorter than the text as it contains less information. This is covered by three numerical features measuring the length of the text, of the hypothesis and the relative length of hypothesis with regard to the text.

3 Deep Semantic Analysis

3.1 Semantic Interpretation

We use a robust wide-coverage CCG-parser (Bos et al., 2004) to generate fine-grained semantic representations for each T/H-pair. The semantic representation language is a first-order fragment of the DRS-language used in Discourse Representation Theory (Kamp and Reyle, 1993), conveying argument structure with a neo-Davidsonian analysis and including the recursive DRS structure to cover negation, disjunction, and implication. Consider for example:

Example: 78 (FALSE)
T: Clinton's new book is not big seller here.
H: Clinton's book is a big seller.



Proper names and definite descriptions are treated as anaphoric, and bound to previously introduced discourse referents if possible, otherwise accommodated. Some lexical items are specified as presupposition triggers. An example is the adjective 'new' which has a presuppositional reading, as shown by the existence of two different "book" entities in drs(T). Scope is fully specified.

To check whether an entailment holds or not, we use two kinds of automated reasoning tools: Vampire, a theorem prover (Riazanov and Voronkov, 2002), and Paradox, a model builder (Claessen and Sörensson, 2003). Both tools are developed to deal with inference problems stated in first-order logic. We use the standard translation from DRS to first-order logic (Kamp and Reyle, 1993) to map our semantic representation onto the format required by the inference tools.

3.2 Theorem Proving

Given a T/H pair, a theorem prover can be used to find answers to the following conjectures:

1. T implies H (shows entailment)
2. T+H are inconsistent (shows no entailment)

Assume that the function DRS denotes the DRS corresponding to T or H, and FOL the function that translates a DRS into first-order logic. Then, if the theorem prover manages to find a proof for

$$\text{FOL}(\text{DRS}(T)) \rightarrow \text{FOL}(\text{DRS}(H)) \qquad \text{(A)}$$

we know that we are dealing with a true entailment. In addition, to use a theorem prover to detect inconsistencies in a T/H pair, we give it:

$$\neg\text{FOL}(\text{DRS}(T);\text{DRS}(H)) \qquad \text{(B)}$$

If the theorem prover returns a proof for (B), we know that T and H are inconsistent and T definitely doesn't entail H (assuming that T and H are themselves consistent).

Examples The theorem prover will find that T implies H for the following examples:

Example: 1005 (TRUE)
T: Jessica Litman, a law professor at Michigan's Wayne State University, has specialized in copyright law and Internet law for more than 20 years.
H: Jessica Litman is a law professor.

Example: 1977 (TRUE)
T: His family has steadfastly denied the charges.
H: The charges were denied by his family.

Example: 898 (TRUE)
T: After the war the city was briefly occupied by the Allies and then was returned to the Dutch.
H: After the war, the city was returned to the Dutch.

Example: 1952 (TRUE)
T: Crude oil prices soared to record levels.
H: Crude oil prices rise.

These examples show how deep semantic analysis deals effectively with apposition, active-passive alternation, coordination, and can integrate lexical knowledge.

The RTE dataset only contains a few inconsistent T/H pairs. Even although Example 78 might look like a case in point, it is not inconsistent: It would be if the T in the example would have been *Clinton's new book is not a big seller*. The addition of the adverb *here* makes T+H consistent.

3.3 Background Knowledge

The theorem prover needs background knowledge to support its proofs. Finding a proof for Example 1952 above is only possible if the theorem prover knows that soaring is a way of rising.

How does it know this? Because in addition to the information from T and H alone, we also supply relevant background knowledge in the form of first-order axioms. Instead of giving just FOL(DRS(T);DRS(H)) to the theorem prover, we supply it with (BK $\wedge$ FOL(DRS(T);DRS(H))) where BK is short for the relevant background knowledge.

We generate background knowledge using three kinds of sources: generic knowledge, lexical knowledge, and geographical knowledge. Axioms for generic knowledge cover the semantics of possessives, active-passive alternation, and spatial knowledge. There are about 20 different axioms in the current system and these are the only manually generated axioms. An example is

$$\forall e \forall x \forall y(\text{event}(e) \wedge \text{agent}(e,x) \wedge \text{in}(e,y) \rightarrow \text{in}(x,y))$$

which states that if an event is located in y, then so is the agent of that event.

Lexical knowledge is created automatically from WordNet. A hyponymy relation between two synsets A and B is converted into ∀x(A(x)→B(x)). Two synset sisters A and B are translated into ∀x(A(x)→ ¬B(x)). Here the predicate symbols from the DRS are mapped to WordNet synsets using a variant of Lesk's WSD algorithm (Manning and Schuetze, 1999). Examples 78 and 1952 would be supported by knowledge similar to:

∀x(clinton(x)→person(x))	∀x(book(x)→artifact(x))
∀x(artifact(x)→ ¬person(x))	∀x(soar(x)→rise(x))

Finally, axioms covering geographical knowledge about capitals, countries and US states are extracted automatically from the CIA factbook. An example:

∀x∀y(paris(x)∧france(y)→in(x,y))

3.4 Model Building

While theorem provers are designed to prove that a formula *is* a theorem (i.e., that the formula is true in any model), they are generally not good at deciding that a formula is *not* a theorem. Model builders are designed to show that a formula is true in at least one model. To exploit these complementary approaches to inference, we use both a theorem prover and a model builder for any inference problem: the theorem prover attempts to prove the input whereas the model builder simultaneously tries to find a model for the negation of the input. If the model builder finds a model for

$$\neg\text{FOL}(\text{DRS}(T))\rightarrow\text{FOL}(\text{DRS}(H)) \qquad (= \neg A)$$

we know that there can't be a proof for its negation (hence no entailment). And if the model builder is able to generate a model for

$$\text{FOL}(\text{DRS}(T);\text{DRS}(H)) \qquad (= \neg B)$$

we know that T and H are consistent (maybe entailment). (In practice, this is also a good way to terminate the search for proofs or models: if the theorem prover finds a proof for $\neg\phi$, we can halt the model builder to try and find a model for ϕ (because there won't be one), and vice versa.)

Another attractive property of a model builder is that it outputs a model for its input formula (only of course if the input is satisfiable). A model is here the logical notion of a model, describing a situation in which the input formula is true. Formally, a model is a pair $\langle D, F\rangle$ where D is the set of entities in the domain, and F a function mapping predicate symbols to sets of domain members. For instance, the model returned for fol(drs(T)) in Example 78 is one where the domain consists of three entities (domain size = 3):

```
D = {d1,d2,d3}      F(loc) = {}
F(book) = {d1,d2}   F(seller) = {}
F(clinton) = {d3}   F(be) = {}
F(of) = {(d1,d3)}   F(agent) = {}
F(big) = {}         F(patient) = {}
```

Model builders like Paradox generate finite models by iteration. They attempt to create a model for domain size 1. If they fail, they increase the domain size and try again, until either they find a model or their resources run out. Thus, although there are infinitely many models satisfying fol(drs(T)), model builders generally build a model with a minimal domain size. (For more information on model building consult (Blackburn and Bos, 2005)).

3.5 Approximating Entailment

In an ideal world we calculate all the required background knowledge and by either finding a proof or a countermodel, decide how T and H relate with respect to entailment. However, it is extremely hard to acquire all the required background knowledge. This is partly due to the limitations of word sense disambiguation, the lack of resources like WordNet, and the lack of general knowledge in a form suitable for automatic inference tasks.

To introduce an element of robustness into our approach, we use the models as produced by the model builders to measure the "distance" from an entailment. The intuition behind it is as follows. If H is entailed by T, the model for T+H is not informative compared to the one for T, and hence does not introduce new entities. Put differently, the domain size for T+H would equal the domain size of T. In contrast, if T does not entail H, H normally introduce some new information (except when it contains negated information), and this will be reflected in the domain size of T+H, which then is larger than the domain size of T. It turns out that this difference between the domain sizes is a useful way of measuring the likelihood of entailment. Large differences are mostly not entailments, small differences mostly are. Consider the following example:

Example: 1049 (TRUE)

T: Four Venezuelan firefighters who were traveling to a training course in Texas were killed when their sport utility vehicle drifted onto the shoulder of a highway and struck a parked truck.

H: Four firefighters were killed in a car accident.

Although this example is judged as a true entailment, Vampire doesn't find a proof because it lacks the background knowledge that one way of causing a car accident is to drift onto the shoulder of the highway and strike something. It generates a model with domain size 11 for fol(drs(T)), and a model with domain size 12 for fol((drs(T);drs(H))). The absolute difference in domain sizes is small, and therefore likely to indicate an entailment. Apart from the absolute difference we also compute the difference relative to the domain size. For the example above the relative domain size yields $1/12 = 0.083$.

The domain size only tells us something about the number of entities used in a model—not about the number of established relations between the model's entities. Therefore, we also introduce the notion of model size. The model size is defined here by counting the number of all instances of two-place relations (and three-place relations, if there are any) in the model, and multiplying this with the domain size. For instance, the following model

```
D = {d1,d2,d3}
F(cat) = {d1,d2}
F(john) = {d3}
F(of) = {(d1,d3)}
F(like) = {(d3,d1),(d3,d2)}
```

has a domain size of 3 and 3 instantiated two-place relations, yielding a model size of $3 * 3 = 9$.

3.6 Deep Semantic Features

Given our approach to deep semantic analysis, we identified eight features relevant for recognising textual entailment. The theorem prover provides us with two features: `entailed` determining whether T implies H, and `inconsistent` determining whether T together with H is inconsistent. The model builder gives us six features: `domainsize` and `modelsize` for T+H as well as the absolute and relative difference between the sizes of T and T+H, both for the size of the domains (`domainsizeabsdif`, `domainsizereldif`) and the size of the models (`modelsizeabsdif`, `modelsizereldif`).

4 Experiments

There are not many test suites available for textual inference. We use throughout this section the dataset made available as part of the RTE challenge.

4.1 Dataset Design and Evaluation Measures

The organisers released a development set of 567 sentence pairs and a test set of 800 sentence pairs. In both sets, 50% of the sentence pairs were annotated as TRUE and 50% as FALSE, leading to a 50% most frequent class baseline for automatic systems. The examples are further distinguished according to the way they were designed via a so-called *Task* variable. For examples marked CD (Comparable Documents), sentences with high lexical overlap in comparable news articles were selected, whereas the hypotheses of examples marked QA (Question Answering) were formed by translating questions from e.g., TREC into statements. The other subsets are IE (Information extraction), MT (Machine Translation) RC (Reading Comprehension), PP (Paraphrase Acquisition) and IR (Information Retrieval). The different examples and subsets cover a wide variety of different aspects of entailment, from incorporation of background knowledge to lexical to syntactic entailment and combinations of all these. For a more exhaustive description of dataset design we refer the reader to (Dagan et al., 2005).

4.2 Experiment 1: Human Upper bound

To establish a human upper bound as well as investigate the validity of the datasets issued, one of the authors annotated all 800 examples of the test set for entailment, using the short RTE annotation rules. The annotation was performed before the release of the gold standard annotation for the test set and was therefore independent of the organisers' annotation. The organisers' and the author's annotation yielded a high percentage agreement of 95.25%. However, 33% of the originally created examples were already filtered out of the corpus before release by the organisers because of agreement-related problems. Therefore we expect that human agreement on textual entailment in general is rather lower.

4.3 Decision trees for entailment recognition

We expressed each example pair as a feature vector, using different subsets of the features described in Section 2 and Section 3 for each experiment. We then trained a decision tree for classification into TRUE and FALSE entailment on the development set, using the Weka machine learning tool (Witten and Frank, 2000), and tested on the test set. Apart from a classification, Weka also computes a confidence value for each decision, dependent on the leaf in the tree that the classified example falls into: if the leaf covers x examples in the training set, of which y examples are classified wrongly, then the error rate is y/x and the confidence value is $1 - y/x$.

Our evaluation measures are accuracy (acc) as the percentage of correct judgements as well as confidence-weighted average score (cws), which rewards the system's ability to assign a higher confidence score to correct judgements than wrong ones (Dagan et al., 2005): after the n judgements are sorted in decreasing order by their confidence value, the following measure is computed:

$$cws = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{\#correct-up-rank-}i}{i}$$

All evaluation measures are computed over the whole test set as well as on the 7 different subsets (CD, IE, etc.). The results are summarised in Table 1. We also computed precision, recall and F-measure for both classes TRUE and FALSE and will discuss the results in the text whenever of interest.

Experiment 2: Shallow Features In this experiment only the shallow features (see Section 2) were used. The overall accuracy of 56.9% is significantly higher than the baseline.[2]

Column 2 in Table 1 shows that this decent performance is entirely due to excellent performance on the CD subset. (Recall that the CD set was designed explicitly with examples with high lexical overlap in mind.) In addition, the method overestimates the number of true entailments, achieving a Recall of 0.926 for the class TRUE, but a precision of only 0.547 on the same class. In contrast, it has good precision (0.761) but low recall (0.236) for the FALSE class. Thus, there is a correspondence between low word overlap and FALSE examples (see Example 731 in the Introduction, where important words in the hypothesis like "swamps" or "marshes" are not matched in the text); high overlap, however, is normally necessary but not sufficient for TRUE entailment (see also Example 78 in Section 3).

Experiment 3: Strict entailment To test the potential of entailment as discovered by theorem proving alone, we now use only the `entailment` and `inconsistent` features. As to be expected, the decision tree shows that, if a proof for T implies H has been found, the example should be classified as TRUE, otherwise as FALSE.[3] The precision (0.767) for the class TRUE is reasonably high: if a proof is found, then an entailment is indeed very likely. However, recall is very low (0.058) as only 30 proofs were found on the test set (for some examples see Section 3). This yields an F-measure of only 0.10 for the TRUE class. Due to the low recall, the overall accuracy of the system (0.52, see Table 1) is not significantly higher than the baseline.

Thus, this feature behaves in the opposite way to shallow lexical overlap and overgenerates the FALSE class. Missing lexical and background knowledge is the major cause for missing proofs.

Experiment 4: Approximating entailment As discussed in Section 3.5 we now try to compensate for missing knowledge and improve recall for TRUE entailments by approximating entailment with the features that are furnished by the model builder. Thus, Experiment 4 uses all eight deep semantic analysis features, including the features capturing differences in domain- and modelsizes. The recall for the TRUE class indeed jumps to 0.735. Although, unavoidably, the FALSE class suffers, the resulting overall accuracy (0.562, see Column 4 in Table 1) is significantly higher than when using the features provided by the theorem prover alone (as in Experiment 3). The confidence weighted score also rises substantially from 0.548 to 0.608. The approximation achieved can be seen in the different treatment of Example 1049 (see Section 3.5) in Experiments 3 and 4. In Experiment 3, this example

[2]We used the z-test for the difference between two proportions to measure whether the difference in accuracy between two algorithms or an algorithm and the baseline is statistically significant at the 5% level.

[3]The `inconsistent` feature was not used by the decision tree as very few examples were covered by that feature.

Table 1: Summary of Results for Experiments 1 to 6

Exp	1: Human		2: Shallow		3: Strict		4: Deep		5: Hybrid		6: Hybrid+Task	
Task	acc	cws	acc	cws	acc	cws	acc	cws	acc	cws	acc	cws
CD	0.967	n/a	**0.827**	**0.881**	0.547	0.617	0.713	0.787	0.700	0.790	**0.827**	0.827
IE	0.975	n/a	0.508	0.503	**0.542**	0.622	0.533	0.616	**0.542**	0.639	**0.542**	**0.627**
MT	0.900	n/a	0.500	0.515	0.500	0.436	**0.592**	**0.596**	0.525	0.512	0.533	0.581
QA	0.961	n/a	0.531	0.557	0.461	0.422	0.515	0.419	0.569	0.520	**0.577**	**0.531**
RC	0.979	n/a	0.507	0.502	**0.557**	0.638	0.457	0.537	0.507	0.587	**0.557**	**0.644**
PP	0.920	n/a	0.480	0.467	0.540	0.581	0.520	0.616	0.560	**0.667**	**0.580**	0.619
IR	0.922	n/a	0.511	0.561	0.489	0.421	0.567	0.503	**0.622**	**0.569**	0.611	0.561
all	0.951	n/a	0.569	0.624	0.520	0.548	0.562	0.608	0.577	0.632	**0.612**	**0.646**

is wrongly classified as FALSE as no proof can be found; in Experiment 4, it is correctly classified as TRUE due to the small difference between domain- and modelsizes for T and T+H.

There is hardly any overall difference in accuracy between the shallow and the deep classifier. However, it seems that the shallow classifier in its current form has very little potential outside of the CD subset whereas the deep classifier shows a more promising performance for several subsets.

Experiment 5: Hybrid classification As shallow and deep classifiers seem to perform differently on differently designed datasets, we hypothesized that a combination of these classifiers should bring further improvement. Experiment 5 therefore used all shallow and deep features together. However, the overall performance of this classifier (see Column 5 in Table 1) is not significantly better than either of the separate classifiers. Closer inspection of the results reveals that, in comparison to the shallow classifier, the hybrid classifier performs better or equally on all subsets but CD. In comparison to the deep classifier in Column 4, the hybrid classifier performs equally well or better on all subsets apart from MT. Overall, this means more robust performance of the hybrid classifier over differently designed datasets and therefore more independence from dataset design.

Experiment 6: Dependency on dataset design As Eperiment 5 shows, simple combination of methods, while maybe more robust, will not necessarily raise overall performance if the system does not know when to apply which method. To test this hypothesis further we integrated the subset indicator as a feature with the values CD, IE, MT, RC, IR, PP, QA into our hybrid system. Indeed, the resulting overall accuracy (0.612) is significantly better than either shallow or deep system alone. Note that using both a combination of methodologies *and* the subset indicator is necessary to improve on individual shallow and deep classifiers for this corpus. We integrated the subset indicator also into the shallow and deep classifier by themselves, yielding classifiers Shallow+Task and Deep+Task, with no or only very small changes in accuracy (these figures are not included in Table 1).

5 Related Work

Our shallow analysis is similar to the IDF models proposed by (Monz and de Rijke, 2003; Saggion et al., 2004). We have expanded their approach by using other shallow features regarding text length.

The basic idea of our deep analysis, using a detailed semantic analysis and first-order inference, goes back to (Blackburn and Bos, 2005). It is similar to some of the recent approaches that were proposed in the context of the PASCAL RTE workshop, i.e. using the OTTER theorem prover (Akhmatova, 2005; Fowler et al., 2005), using EPILOG (Bayer et al., 2005), or abduction (Raina et al., 2005).

None of these systems, however, incorporate model building as a central part of the inference mechanism. We have shown that solely relying on theorem proving is normally insufficient due to low recall, and that using model builders is a promising way to approximate entailment.

Results of other approaches to determining textual entailment indicate that it is an extremely hard task. The aforementioned RTE workshop revealed that participating systems reached accuracy figures ranging between 0.50 and 0.59 and cws scores between 0.50 and 0.69 (Dagan et al., 2005). Comparing this with our own results (accuracy 0.61 and cws 0.65) shows how well our systems performs on the same data set. This is partly due to our hybrid approach which is more robust across different datasets.

6 Conclusions

Relying on theorem proving as a technique for determining textual entailment yielded high precision but low recall due to a general lack of appropriate background knowledge. We used model building as an innovative technique to surmount this problem to a certain extent. Still, it will be unavoidable to incorporate automatic methods for knowledge acquisition to increase the performance of our approach. Future work will be directed to the acquisition of targeted paraphrases that can be converted into background knowledge in the form of axioms.

Our hybrid approach combines shallow analysis with both theorem proving and model building and achieves high accuracy scores on the RTE dataset compared to other systems that we are aware of. The results for this approach also indicate that (a) the choice of entailment recognition methods might have to vary according to the dataset design and/or application and (b) that a method that wants to achieve robust performance across different datasets might need the integration of several different entailment recognition methods as well as an indicator of design methodology or application.

Thus, although test suites establish a controlled way of assessing textual entailment detection systems, the importance of being able to predict textual entailment in NLP might be better justified using task-based evaluation. This can be achieved by incorporating them in QA or summarisation systems.

Acknowledgements We would like to thank Mirella Lapata and Malvina Nissim as well as three anonymous reviewers for their comments on this paper. We are also grateful to Valentin Jijkoun and Bonnie Webber for discussion and Steve Clark and James Curran for help on using the CCG-parser.

References

E. Akhmatova. 2005. Textual entailment resolution via atomic propositions. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

S. Bayer, J. Burger, L. Ferro, J. Henderson, and A. Yeh. 2005. Mitre's submission to the eu pascal rte challenge. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

P. Blackburn and J. Bos. 2005. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI.

J. Bos, S. Clark, M. Steedman, J. Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proc of COLING*.

K. Claessen and N. Sörensson. 2003. New techniques that improve mace-style model finding. In *Model Computationa - Principles, Algorithms, Applications (Cade-19 Workshop)*, Miami, Florida.

I. Dagan, O. Glickman, and B. Magnini. 2005. The pascal recognising textual entailment challenge. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

A. Fowler, B. Hauser, D. Hodges, I. Niles, A. Novischi, and J. Stephan. 2005. Applying cogex to recognize textual entailment. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht, Netherlands.

C. Manning and H. Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

C. Monz and M. de Rijke. 2003. Light-weight entailment checking for computational semantics. In *Proc. of ICOS-3*.

R. Raina, A.Y. Ng, and C. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proc. of AAAI 2005*.

A. Riazanov and A. Voronkov. 2002. The design and implementation of Vampire. *AI Comm.*, 15(2-3).

H. Saggion, R. Gaizauskas, M. Hepple, I. Roberts, and M Greenwood. 2004. Exploring the performance of boolean retrieval strategies for open domain question answering. In *Proc. of the IR4QA Workshop at SIGIR*.

I. H. Witten and E. Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Diego, CA.

A Self-Learning Context-Aware Lemmatizer for German

Praharshana Perera and **René Witte**
Institute for Program Structures and Data Organization (IPD)
Universität Karlsruhe, Germany
perera|witte@ipd.uka.de

Abstract

Accurate lemmatization of German nouns mandates the use of a lexicon. Comprehensive lexicons, however, are expensive to build and maintain. We present a self-learning lemmatizer capable of automatically creating a full-form lexicon by processing German documents.

1 Introduction

Lemmatization is the process of deriving the base form, or *lemma*, of a word from one of its inflected forms. For morphologically complex languages like German this is not a simple task that can be solved solely through a rule-based algorithm: Performing an accurate lemmatization for German requires a lexicon. This can be either a lexicon containing all inflected forms of a word together with its base form *(full-form lexicon)*, or just the lemma together with a set of rules for creating its inflected forms *(base-form lexicon)* (Hausser, 2000).

Creating such a lexicon by hand, however, is expensive and time-consuming. Perhaps because of this there are currently no freely available lexical resources for German that include full case and inflection information.[1] Moreover, even a full-scale commercial lexicon can fail when encountering specialized terminology.

As a consequence, most systems processing German texts currently perform the much simpler task of *stemming*, which often generates stem forms of words that might not actually exist in the language (so-called *overstemming*). Stemming is frequently used for information retrieval (IR) tasks, an example being the German stemmer contained in the full-text search engine *Lucene*,[2] which is based on the algorithm described in (Caumanns, 1999). While overstemming is a feasible approach for text retrieval, a *text mining* system often needs to obtain a more precise lemma, for example, in order to perform a gazetteer lookup to identify named entities or for description logic (DL) queries within an ontology.

The goal of our work, therefore, is to allow the semi-automatic generation of a lexicon by mining full-text documents. Since there are currently no free lemmatization systems for German available,[3] all components have been developed for release as free, open-source software.

2 Lemmatization Algorithm

Our lemmatization system has two main components, an algorithm and a lexicon. The algorithm lemmatizes German nouns depending on morphological classes. The lexicon, which is described in Section 3, is generated from the nouns that have been processed by this algorithm, with some additional capabilities for self-correction.

The lemmatization algorithm considers the context and grammatical features of the language to lemmatize German words. It requires an additional POS tagger and an NP chunker, which are used as resources to extract the features of words and their surrounding context. It has been developed primarily for nouns but can also be extended to lemmatize adjectives and verbs.

2.1 Inflection of German Nouns

In German there are seven declensional suffixes for nouns: *-s*, *-es*, *-e*, *-n*, *-er*, and *-ern* (Caumanns, 1999). These suffixes are due to the morphological

[1]The free online dictionary *Wiktionary* (http://de.wiktionary.org/) had at the time of writing (May 2005) less than 5000 entries for German.

[2]http://lucene.apache.org/

[3]The *Morphy* system (Lezius et al., 1998) is described as "freely available," but in fact is closed-source, binary-only, nonchangeable software. It is also no longer being maintained.

Class	Features	Remove Suffix
I	$\{Sg\} \wedge \sim \{Gen\}$	
	$\wedge\{Masc \vee Fem \vee Neut\}$	none
II	$\{Sg\} \wedge \{Gen\}$	
	$\wedge\{Masc \vee Neut\}$	*-es* or *-s*
III	$\{Pl\} \wedge \sim \{Dat\}$	
	$\wedge\{Masc \vee Fem \vee Neut\}$	*-e*, *-n*, *-en*, *-er*, or *-s*
IV	$\{Pl\} \wedge \{Dat\}$	
	$\wedge\{Masc \vee Fem \vee Neut\}$	*-n*, *-en*, *-ern*, or *-s*

Table 1: Lemmatization of German nouns based on morphological classes

features such as gender, number, and case (Vilares et al., 2004). A basic lemmatization algorithm would reduce the suffixes by analyzing these morphological features. The existence of these suffixes is caused by the following: (1) genitive form of the singular, masculine, or neuter nouns have the declensional suffixes *-es*, *-en*, or *-s*, e.g., *Kind → Kindes*; (2) plural nouns have the declensional suffixes *-en*, *-ern*, *-n*, or *-s*, e.g., *Frau → Frauen*; and (3) dative forms of plural nouns have the declensional suffixes *-s*, *-n*, *-en*, or *-ern*, like in *Kind → Kindern*.

A simple lemmatization algorithm has been developed to cutoff these suffixes taking the morphological features such as number, gender, and case into consideration. The values of these features often cannot be uniquely determined from the word form (Evert, 2004). Therefore, we developed an algorithm to classify the nouns into four different morphological classes, as shown in Table 1. Lemmatization can then be performed based on these morphological classes (Table 1, right column).

We now discuss the first step, finding the proper class for each noun.

2.2 Lemmatization Classes

The currently available POS taggers for German do not capture more complex morphological features like number or case. Thus, in order to lemmatize German nouns it is necessary to first categorize them into the classes defined above. Our algorithm achieves this by analyzing the grammatical features of a noun, based on the German grammar (Duden, 1995). Additionally, a stochastic case tagger has been developed as an additional resource to support the algorithm in the classification of nouns.

2.2.1 Nouns with a Determiner

Table 2 shows statistics for German noun phrases for different corpora (the size of each corpus can be found in Section 5). The percentage of nouns that have a determiner is around 34% (25% determiner only + 9% determiner and modifier). The morphological information that can be extracted from a determiner preceding a noun is very ambiguous. For example, the determiner *die* can be either singular or plural in number, nominative or accusative in case, and masculine, feminine, or neuter in gender. But some determiners can be used to classify nouns into morphological classes.

Corpus	Det Only	Mod Only	Det+Mod	None
Negra	25%	13%	9%	53%
Die Welt	26%	14%	9%	51%
AvFIS	22%	16%	8%	53%
Wikipedia	28%	15%	9%	48%

Table 2: Distribution of German noun phrases

Table 3 describes our algorithm for nouns that have a determiner. In the first step, we consider determiners that are singular and non-genitive. Therefore, they belong to class I and do not need to be lemmatized. Examples are *das Haus → Haus*, *dem Mann → Mann*, *eine Frau → Frau*.

Determiners in the second step are singular and genitive and the gender can be masculine or neuter. These nouns belong to class II and to find the lemma, the suffix *-s* or *-es* must be removed. Examples are *des Hauses → Haus*, *des Vaters → Vater*.

Determiners in the third step can be either singular or plural. The only possible way to differentiate this is when the noun has both a determiner and a modifier. The plurals have modifiers ending with *-en* and singulars with *-e*.

In the other steps, nouns cannot be directly classified. In the fourth step we apply additional heuristics and in the last step the statistical case tagger (described in Section 2.4) is being used.

In German, genitive is mostly used as the case of nominal modifiers and complement of prepositions (Hinrichs and Trushkina, 1996), which is used as a heuristic to find the singular determiners in the fourth step and in the same way another heuristic has been applied which finds singular determiners when they are followed by dative prepositions.

The determiner *den* in German can be either accusative or dative. In the dative case it is plural and in the accusative case it is singular and masculine in gender. Examples are *den Kindern* (dative plural)

Step	Determiner	Class
1	das, dem, ein, einem, ..., ihr, ihrem	Class I
2	des, eines, meines, deines, ..., ihres	Class II
3	die, meine, deine, ..., ihre	If modifier has the suffix *-e* → Class I If modifier has the suffix *-en* → Class III
4	der, meiner, deiner, ..., ihrer	If determiner is not followed by a genitive preposition or a noun phrase → Class I If determiner is followed by a dative preposition → Class I
5	den, meinen, deinen, ..., ihren	If case tagged by case tagger is accusative → Class I If case is dative → Class IV

Table 3: Lemmatizing German nouns that appear with a determiner

and *den Salat* (accusative singular). The fifth step has determiners that have this ambiguity, which is resolved using information given by the case tagger.

2.2.2 Nouns with a Modifier only

The morphological features of a noun that can be extracted from a modifier are less than those based on a determiner. According to the statistics in Table 2, around 14% of noun phrases come with a modifier only. However, it is sometimes possible to lemmatize nouns by looking at the modifiers' suffixes and the case information as given by the case tagger. Table 4 describes our algorithm for nouns that come solely with a modifier.

In German, when a noun exists without a determiner but with a modifier, the ending of the modifier changes according to the morphological features of the noun. For example, the noun phrase *dem kleinen Kind* without determiner becomes *kleinem Kind*. The suffix *-em* appears only for singular nouns, which do not need to be lemmatized.

A modifier with the suffix *-es* can be genitive, accusative, or nominative. A good example for this feature is *kleines Kind* and *kleines Kindes*. In the first case it is nominative or accusative and in the second case genitive. Here, we use the case information given by the case tagger to classify the noun.

Modifiers with the suffix *-en* are similar to the step with the determiner *den*. A modifier with suffix *-en* can be either singular or plural. In singular case it is accusative and in plural case dative; examples

Step	Modifier Suffix	Action
1	*-em*	Class I
2	*-es*	If case is not genitive → Class I If case is genitive → Class II
3	*-en*	If case is accusative → Class I If case is dative → Class IV
4	*-er*	If case is dative or nominative → Class I

Table 4: Lemmatizing German nouns with a modifier but without a determiner

for these cases are *guten Mann* (accusative, singular) and *guten Männern* (dative, plural).

Modifiers that have the suffix *-er* can be both genitive or non-genitive. In the non-genitive case they are singular and need not to be lemmatized. Examples for this are *kleiner Katze* (dative, singular), *kleiner Katze* (genitive, singular), and *kleiner Katzen* (genitive, plural).

2.2.3 Nouns without Modifier or Determiner

Nouns without modifier or determiner account for 51% of all NPs (Table 2). Most of these nouns cannot be directly lemmatized using methods as they have been applied above. The main reason for this is the unavailability of a tagger providing number and gender information for such nouns. Using only the case tagger it is not possible to classify all the nouns in this set. However, it is possible to capture some nouns in this set by applying a heuristic:

If a noun follows the preposition *zum, zur, am, im, ins,* or *ans* ⟶ Class I.

The main idea behind this heuristic is a grammatical feature of the German language. In German, there exists a set of prepositions that are connected with a determiner, for example, *zum Bahnhof*, *zur Party*, and *ins Bett*. The main feature of nouns following such a preposition is that they are singular and thus do not need to be lemmatized.

2.2.4 POS-based Lemmatization

To maximize the number of nouns that can be lemmatized a heuristic has been added to capture nominative nouns, using the POS tagger *TreeTagger* (Schmid, 1995). The main idea behind this heuristic is to find the subject and main verb of a sentence. In German, the subject is always nominative and by looking at the suffix of the main verb, it is possible to determine the number of the subject.

This heuristic first finds the subject of the sentence based on the case tagger information. Then, based on the information from the POS tags the main verb is identified and checked whether it is a plural verb. The corresponding plural nouns are then lemmatized, whereas singular nouns remain unchanged.

2.3 Optimizations

To avoid some errors in the lemmatization algorithm and to increase the accuracy of lemmatization additional optimizations are needed. In German, many plural forms are built by changing a vowel to an Umlaut (Caumanns, 1999), like in *das Land* and *die Länder*. But this is not a static rule because there are some cases where the noun already has an Umlaut, like in *die Affäre* and *die Affären*. Here, it would not be correct to lemmatize *Affären* to **Affare*. As a solution, several possible lemma candidates are generated, for example, *Länder* → **Länd* and *Land*.

Another feature of German are nouns that are made up from adjectives. These nouns have different suffixes when they appear with definite or indefinite determiners and without determiners. An example is the noun *Abgeordnete*; in singular form it can appear in two ways, *der Abgeordnete* and *ein Abgeordneter*. It is also tricky in the dative singular case, where it has three forms, *Abgeordnetem*, *Abgeordneter* and *dem/der/einem/einer Abgeordneten*. Our algorithm thus generates the possible lemma candidates: *Abgeordneter* → *Abgeordneter*, *Abgeordnete*.

The main reason to generate lemma candidates for these nouns above is to store them in the lexicon. The correct lemma can then later be identified and the lexicon updated when the noun appears again in a different context.

2.4 The Case Tagger

As an additional resource to the lemmatizer we developed a stochastic case tagger. It has been built using the POS tags as features to train the model in order to predict the case of nouns. From the standard STTS tagset for German (Schiller et al., 1995), which has 54 POS tags, 38 tags[4] have been identified to train the model, based on an analysis of the grammatical structure of German sentences as defined in the German grammar (Duden, 1995).

[4]These POS tags define the structure of the grammatical case in German sentences, for example, verbs and prepositions.

2.4.1 Model

We apply a standard Hidden Markov Model (HMM), designed for the structure of the German language. A German sentence can be represented as a set of variable states, which can be nominative, accusative, dative, or genitive and a set of fixed states like finite verbs and conjunctions. For example, in the sentence *Die Mutter gibt den kleinen Kindern den Salat*, the phrases *Die Mutter* (nominative), *den kleinen Kindern* (dative) and *den Salat* (accusative) are the variable states and the finite verb *gibt* is a fixed state. In this manner, the whole sentence can be represented with the state sequence *nominative VVFIN* (finite verb) *dative accusative*. From the 38 tags that have been chosen for training, 10 tags[5] have been integrated with the nouns as variable states.

2.4.2 Tagging Algorithm

As an HMM tagger, our case tagger chooses the best sequence of tags for a given sequence of states (Jurafsky and Martin, 2000). In this model this can be expressed as choosing the best sequence of tags for the variable states in the sequence. The first stage of the algorithm selects the set of tags from the POS tags that are used for calculation and then it orders these tags into fixed and non-fixed states with respect to the grammatical case. The second stage of the algorithm calculates the most probable tag sequence using the Viterbi algorithm. The model is smoothed to avoid zero probabilities. In the worst case the complexity of this algorithm is $O(N^3)$ but here $N = 4$, the four grammatical cases.

3 Lexicon Generation

As discussed above, the lemmatization algorithm cannot be used alone to lemmatize all German nouns, as it cannot capture every noun in a text. However, a noun that could not be lemmatized within one text may well have enough context information for a precise lemmatization within another. Thus, our main idea here is to create a self-learning lexicon that evolves with the nouns processed by the algorithm, continuously learning the correct values for each lexical entry.

[5]Like for nouns, grammatical case is a morphological feature of these POS tags, for example, pronouns and adjectives.

3.1 Lexicon Entries

The lexicon stores the full form of a word with its base form and possible morphological features like number, gender, and case. This is different from a lexicon as it has been used for lemmatization, which only stores the base form for each word together with its inflection class (Lezius et al., 1998).

For example, the lexicon entries for the noun *Kind* are represented as:

Noun	Number	Gender	Case	Lemma
Kind	Sg	Neut	Nom.Akk	Kind
Kindes	Sg	Neut	Gen	Kind
Kinder	Pl	Neut	Nom.Akk	Kind
Kindern	Pl	Neut	Dat	Kind

3.2 Lexicon Generation

The lexicon grows by updating itself from the nouns that have been processed by the lemmatization algorithm. Additional functionality has been implemented in the lexicon, to allow it to evolve by assigning the correct lemma to the words that are inflected from the same lemma and correcting some errors that have been generated by the algorithm.

3.2.1 Evolving the Lexicon

If a word is scheduled for addition to the lexicon, it first checks whether it already exists. If this is the case, it compares each feature of the new word with the one already in the lexicon. If there is any difference, for example, if the word in the lexicon shows the number *Sg* and the new word has the number *Pl*, it adds both features to the lexicon entry. If a new word does not already exist in the lexicon it will just be added as a new entry. The following example illustrates this process:

Current state of the lexicon				
Menschen	Sg	Masc	Akk	Mensch
Mensch	Sg	Masc	Nom	Mensch

New Entry				
Menschen	Pl	Masc	Nom	Mensche.Mensch

State of the lexicon after update				
Menschen	Sg.Pl	Masc	Akk.Nom	Mensch
Mensch	Sg	Masc	Nom	Mensch

The assignment of the correct lemma *Mensch* is done by a procedure that will be discussed next.

3.2.2 Updating Lemmas

If a new word lemmatized by the algorithm that has more than one lemma candidate is to be added, the lexicon tries to assign the correct lemma for this new word by looking at the lemmas that are already in the lexicon. If one of the lemma candidates in the new word matches with a lemma stored in the lexicon, the lemma of the new word will be updated with the new information. This process is illustrated in the following example:

Current state of the lexicon (lemma only)	
Land	Land
Landes	Land

New Entry	
Länder	Lände.Länd.Lande.Land

State of the lexicon after update	
Land	Land
Landes	Land
Länder	Land

In the same way, if a new word that has been correctly lemmatized is to be entered to the lexicon, the lexicon tries to update the words in the lexicon that have more than one lemma using the lemma of the new word. If one of the lemma candidates of a word in the lexicon matches with the lemma of the new word, then the lemma of the word in the lexicon will be updated with the lemma of the new word:

Current state of the lexicon (lemma only)	
Länder	Lände.Länd.Lande.Land
Ländern	Länder.Lände.Länd.Lander.Lande.Land

New Entry	
Landes	Land

State of the lexicon after update	
Landes	Land
Länder	Land
Ländern	Land

3.2.3 Automatic Error Correction

The lemmatization algorithm may produce errors, for example, a plural noun wrongly tagged as singular may not be lemmatized, resulting in a wrong entry. While the lexicon evolves, such errors produced by the algorithm are corrected automatically.

As shown in the example below, the lexicon can have wrong entries and entering a word with more than one lemma, which is an inflectional form of a word that has a wrong entry, will not be assigned with the correct lemma because the procedure that updates the lemma will assign possible lemma candidates to this word. If a word that has a wrong entry in the lexicon will be entered again with the correct lemma, the word itself and all its inflectional forms will be updated with the correct lemma:

Current state of the lexicon (lemma only)	
Jahr	Jahr
Jahre	Jahre **(wrong)**
New Entry	
Jahren	Jahre.Jahr
State of the lexicon after update	
Jahr	Jahr
Jahre	Jahre **(wrong)**
Jahren	Jahre.Jahr **(two possibilities)**
New Entry	
Jahre	Jahr **(correct lemmatization)**
State of the lexicon after update	
Jahr	Jahr
Jahre	Jahr
Jahren	Jahr

4 Implementation

The lemmatization algorithm and the lexicon have been implemented based on the GATE architecture (Cunningham et al., 2002). GATE provides an infrastructure for developing and deploying software components that process human language. For the German POS tagger we currently use the TreeTagger (Schmid, 1995). The other main resource is a multi-lingual base NP chunker implemented within the JAPE language.

The Negra corpus version 2 (Skut et al., 1998) based on approximately 70 000 tokens tagged with morphological features has been used to train the case tagger. This corpus has been split into 50 000 training tokens and 20 000 tokens used for testing.

5 Evaluation

Evaluation was performed over four collections of texts: (1) a set of 350 articles from “Die Welt” newspaper containing 190 868 tokens (40 104 nouns); (2) the electronic version of the book “AvFIS”[6] containing 120 212 tokens (22 039 nouns); (3) six manually for lemma, case, and number annotated articles from the German *Wikipedia* containing 6580 tokens (1536 nouns); (4) 20 000 tokens (5023 nouns) from the Negra corpus version 2 (Skut et al., 1998), which contains morphological tags for case and number.

The lemmatization of German texts has been evaluated using both the algorithm and the lexicon separately and combined. Since the first two collections of texts are not annotated with lemmatization information, we evaluated the lemma produced by our algorithm or lexicon by comparing it with the one produced by the TreeTagger, which is based on an internal dictionary. Since the TreeTagger cannot produce the lemma for all nouns, we evaluated only that percentage of nouns for which the TreeTagger was able to produced a lemma, which is 88% for both “Die Welt” and the “AvFIS” book. In order to also evaluate our lemmatization independently from the lemma produced by the TreeTagger, we compared its results to a manually annotated set of articles from the Wikipedia.

[6]René Witte, *Architektur von Fuzzy-Informationssystemen*, BoD, 2002, `http://rene-witte.net`

Corpus	Nouns	Algorithm Only		Lexicon Only	
		Lemm.	Acc.	Lemm.	Acc.
Die Welt	35531	49%	0.88	67%	0.96
AvFIS	19394	40%	0.88	70%	0.97
Wikipedia	1536	49%	0.87	54%	0.97

Table 5: Lemmatization results, algorithm and lexicon tested in isolation

Finally, the case and number taggers have also been evaluated separately using the manually annotated articles from the Wikipedia and the Negra corpus. For this evaluation, the lemmatization accuracy has been calculated by $accuracy = \frac{n(correct)}{n(lemmatized)}$.

5.1 Algorithm Evaluation

Table 5 shows the results of lemmatization using only the lemmatization algorithm (i.e., no lexicon).

The number of nouns that our algorithm can lemmatize is just below 50%. This is mainly due to the large number of nouns, as shown in Table 2, that appear without a determiner or modifier, as well as some ambiguous cases where NPs with determiners and modifiers cannot be lemmatized directly.[7]

The accuracy of lemmatization based on this approach shows the irregular morphological features of the German language. 75% of the errors are due to irregular morphological variations in German. The algorithm does not change the vowels with Umlauts, therefore, all nouns which have a vowel with an Umlaut in plural are not lemmatized correctly. For example, the noun *Ländern* is lemmatized by the algorithm to **Länd* but the correct lemma is *Land*. Another peculiarity that causes errors in lemmatization are nouns that have been formed by adjectives. For example, a noun with a determiner like *ein Ab-*

[7]E.g., in the sentence *Ich sehe die Kinder der Frau* the two nouns *Kinder* and *Frau* cannot be lemmatized by the algorithm because in this context these nouns could be singular or plural.

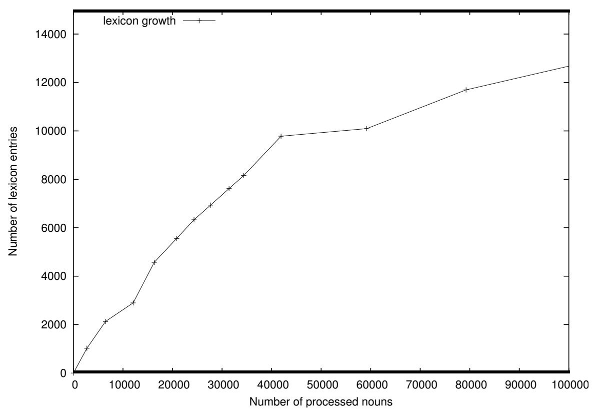


Figure 1: Lexicon growth

geordneter would not be lemmatized by the algorithm because it is singular and non-genitive. However, the correct lemma of this word is *Abgeordnete*. German also has nouns where the plural and the singular forms are equal. This is a situation in which the algorithm fails to generate the correct lemma. For example, the noun *Arbeiter* has the same singular *der Arbeiter* and plural *die Arbeiter* form. The algorithm lemmatizes *die Arbeiter* to **Arbeit* whereas the correct lemma is *Arbeiter*.

The remaining errors are due to mis-tagging, mainly by the case tagger, which can result in an error in lemmatization. For example, *den Kindern* has been tagged by the case tagger as **Akk* (correct *Dat*), so the lemmatization algorithm does not lemmatize this noun to *Kind* because the case is accusative and hence assumed to be singular.

5.2 Lexicon Evaluation

The lexicon was initially generated by applying the lemmatization algorithm on the "Die Welt" collection of texts. We then evaluated lemmatization based solely on the lexicon (not applying the algorithm) for these documents. Table 5 also shows the results for this collection of texts. The growth of the lexicon is shown in Figure 1; when we performed the evaluation it contained 12 858 entries for 10 251 lemmas.

The next test for lexicon evaluation has been done in two stages. First, the electronic book "AvFIS" (2) has been lemmatized using only the lexicon. Afterwards, we applied the lemmatization algorithm on the same book, generating new entries, and then evaluated the extended lexicon again on this book. Before processing the book, the lexicon was able to lemmatize 40% of all nouns with an accuracy of 0.98, whereas afterwards the lemmatization coverage increased to 70% with the accuracy dropping slightly to 0.97.

Both tests above have been done against the lemma generated by the TreeTagger. Additionally, we evaluated the lexicon on our manually annotated set of articles from the Wikipedia, which is also shown in Table 5.

As can be seen, in all tests the accuracy of lemmatization based on the lexicon is higher than that of the algorithm. The reason for this is the self-correcting feature of the lexicon discussed above: While the lexicon evolves it increasingly assigns the correct lemma for each noun.

Although the lexicon performs with a high accuracy, the remaining errors are due to various forms of the construction of words in German. For example, consider the two nouns *Sieger* (lemma *Sieger*) and *Sieg* (lemma *Sieg*). As the lexicon evolves, it assigns *Sieger* the lemma **Sieg* because it already exists as a lemma in the lexicon whereas the correct lemma is *Sieger*. Some remaining incorrect entries in the lexicon also result in errors. Such cases will need to be corrected manually.

The percentage of lemmatization is obviously high for texts which have been used to generate the lexicon. The difference can be clearly seen in the book example, where the number of nouns that could be lemmatized increased significantly after enhancing the lexicon from the same set of nouns.

5.3 Lexicon and Algorithm Evaluation

We evaluated lemmatization using both algorithm and lexicon combined on the same set of texts (Table 6, right side). The number of lemmatized nouns has clearly increased in the combined method. Here, a lemma produced by the lexicon takes precedence over the algorithms' one, if both were able to produce a lemma. Table 6 also shows the contribution of each method for lemmatization in the combined method (left side). The number of nouns lemmatized

Corpus	Contribution			Results	
	Lex.	Alg.	Both	Lemm.	Acc.
Die Welt	27%	10%	39%	76%	0.94
AvFIS	33%	3%	37%	73%	0.96
Wikipedia	24%	19%	30%	73%	0.93

Table 6: Results using both algorithm and lexicon

by the lexicon is relatively higher than the algorithm on the first two texts because these texts were used to initially generate the lexicon.

When both algorithm and lexicon were able to produce a lemma, it agrees in 92% of all cases with an accuracy of 0.98.

One special case both fail to lemmatize correctly are foreign (e.g., Latin) words that do not follow German morphological rules (e.g., *Lexika* → *Lexikon*). These require manual correction or the development of specialized heuristics.

Finally, we evaluated the performance of the case and number taggers. While a detailed discussion of these results cannot be presented in this paper, the case tagger reaches an accuracy of 0.92 on the training data, 0.8 on the testing data, and 0.79 on the Wikipedia, while the number tagger has an accuracy of 0.93 on the training data, 0.9 on the testing data, and 0.91 on the Wikipedia corpus.

6 Conclusions and Future Work

In this paper we demonstrated a new algorithm for the lemmatization of German nouns. An important feature is the automatic construction of a lexicon from the processed documents, allowing it to continuously improve in both coverage and accuracy. The lemmatization system as well as a lexicon will be made available as free, open-source software, which will fill an important gap for the development of NLP systems dealing with German.[8]

The automatic generation and self-correction of a lexicon is a huge time-saver. Compared to the German Wiktionary, whose users needed a year to manually curate less than 5000 entries, we were able to compile the same amount of nouns within a matter of days.[9] Human intervention can be limited to the inspection and correction of wrong entries, which will allow the creation of specialized lexicons even for groups with limited resources. To increase the coverage of our lexicon, we currently employ a web crawler, which daily scans several German news sources for texts, which are then processed for lexical entries.

In the future, we plan to enhance the system to also deal with verbs, adjectives, and adverbs, as well as compound nouns.

Acknowledgments. This work is funded through the DFG project “Entstehungswissen.”

[8]Dictionaries that are only accessible online, like Canoo.net (`http://www.canoo.net`) or Wortschatz Lexikon (`http://wortschatz.uni-leipzig.de`) we do not consider freely available, as the underlying databases and tools cannot be downloaded, modified, or integrated into NLP systems.

[9]The Wiktionary does have more information for each entry, however, some of these could also be automatically created in a similar fashion.

References

Jörg Caumanns. 1999. A Fast and Simple Stemming Algorithm for German Words. Technical report, Center für Digitale Systeme, Freie Universität Berlin.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the ACL*. `http://gate.ac.uk`.

Duden. 1995. *Grammatik der deutschen Gegenwartssprache*. Dudenverlag, Mannheim, 5th edition.

Stefan Evert. 2004. The Statistical Analysis of Morphosyntactic Distributions. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004), Lisbon, Portugal*.

Roland Hausser. 2000. *Grundlagen der Computerlinguistik*. Springer Verlag.

E. Hinrichs and J. Trushkina. 1996. Forging agreement: Morphological disambiguation of noun phrases. In *Proceedings of the First Workshop on Treebanks and Linguistic Theory*.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall PTR.

Wolfgang Lezius, Reinhard Rapp, and Manfred Wettler. 1998. A Freely Available Morphological Analyzer, Disambiguator and Context Sensitive Lemmatizer for German. In *Proc. COLING-ACL*, pages 743–748.

A. Schiller, S. Teufel, and C. Thielen. 1995. Guidelines für das Tagging deutscher Textkorpora mit STTS. Technical report, Universität Stuttgart and Tübingen.

H. Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*.

Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A Linguistically Interpreted Corpus of German Newspaper Text. In *Proceedings of the ESS-LLI Workshop on Recent Advances in Corpus Annotation. Saarbrücken, Germany*.

Jesús Vilares, Miguel A. Alonso, and Manuel Vilares. 2004. Morphological and Syntactic Processing for Text Retrieval. In *DEXA 2004*, Springer LNCS 3180.

A Robust Combination Strategy for Semantic Role Labeling

Lluís Màrquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo
Technical University of Catalunya
Barcelona, Spain
{lluism,surdeanu,pcomas,turmo}@lsi.upc.edu

Abstract

This paper focuses on semantic role labeling using automatically-generated syntactic information. A simple and robust strategy for system combination is presented, which allows to partially recover from input parsing errors and to significantly boost results of individual systems. This combination scheme is also very flexible since the individual systems are not required to provide any information other than their solution. Extensive experimental evaluation in the CoNLL-2005 shared task framework supports our previous claims. The proposed architecture outperforms the best results reported in that evaluation exercise.

1 Introduction

The task of Semantic Role Labeling (SRL), i.e. the process of detecting basic event structures such as *who* did *what* to *whom*, *when* and *where*, has received considerable interest in the past few years (Gildea and Jurafsky, 2002; Surdeanu et al., 2003; Xue and Palmer, 2004; Pradhan et al., 2005a; Carreras and Màrquez, 2005). It was shown that the identification of such event frames has a significant contribution for many Natural Language Processing (NLP) applications such as Information Extraction (Surdeanu et al., 2003) and Question Answering (Narayanan and Harabagiu, 2004).

Most current SRL approaches can be classified in one of two classes: approaches that take advantage of complete syntactic analysis of text, pioneered by Gildea and Jurafsky (2002), and approaches that use partial syntactic analysis, championed by previous evaluations performed within the Conference on Computational Natural Language Learning (CoNLL) (Carreras and Màrquez, 2004). The wisdom extracted from this volume of work indicates that full syntactic analysis has a significant contribution to the SRL performance, *when using hand-corrected syntactic information*.

On the other hand, when only automatically-generated syntax is available, the quality of the information provided through full syntax decreases because the state-of-the-art of full parsing is less robust and performs worse than the tools used for partial syntactic analysis. Under such real-world conditions, the difference between the two SRL approaches (with full or partial syntax) is not that high. More interestingly, *the two SRL strategies perform better for different semantic roles*. For example, models that use full syntax recognize better agent and theme roles, whereas models based on partial syntax are better at recognizing explicit patient roles, which tend to be farther from the predicate and accumulate more parsing errors (Màrquez et al., 2005).

The above observations motivate the work presented in this paper. We introduce a novel semantic role labeling approach that combines several individual SRL systems. Intuitively, our approach can be separated in two stages: a *candidate generation* phase, where the solutions provided by several individual models are combined into a pool of candidate arguments, and an *inference* phase, where the candidates are filtered using a binary classifier, and possi-

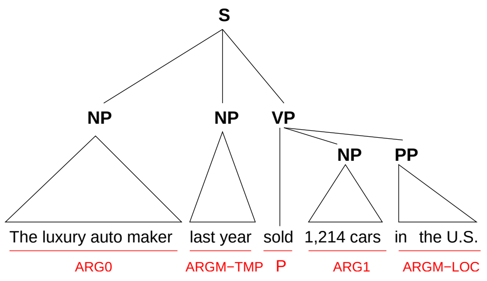


Figure 1: Sample PropBank sentence.

ble conflicts with domain knowledge constraints are resolved to obtain the final solution.

For robustness, the inference model uses only global attributes extracted from the solutions provided by the individual systems, e.g., the sequence of role labels generated by each system for the current predicate. We do not use any attributes specific to the individual models, not even the confidence assigned by the individual classifiers. Besides simplicity, the consequence of this decision is that our approach does not impose any restrictions on the individual SRL strategies, as long as one solution is provided for each predicate. On the other hand, probabilistic inference processes, which have been successfully used for SRL (Koomen et al., 2005), mandate that each individual candidate argument be associated with its raw activation, or confidence, in the given model. However, this information is not directly available in two out of three of our individual models, which classify argument chunks and not entire arguments.

Despite its simplicity, our approach obtains encouraging results: the combined system outperforms any of the individual systems and, using exactly the same data, it is also competitive with the best SRL systems that participated in the latest CoNLL shared task evaluation (Carreras and Màrquez, 2005).

2 Semantic Corpora

In this paper we report results using PropBank, an approximately one-million-word corpus annotated with predicate-argument structures (Kingsbury et al., 2002). To date, PropBank addresses mainly predicates lexicalized by verbs and a small number of predicates lexicalized by verb nominalizations and adjectives.

The arguments of each predicate are numbered sequentially from `ARG0` to `ARG5`. Generally, `ARG0` stands for *agent*, `ARG1` for *theme* or *direct object*, and `ARG2` for *indirect object*, *benefactive* or *instrument*, but mnemonics tend to be verb specific. Additionally, predicates might have "adjunctive arguments", referred to as `ARGM`s. For example, `ARGM-LOC` indicates a locative and `ARGM-TMP` indicates a temporal. Figure 1 shows a sample sentence where one predicate ("sold") has 4 arguments.

In a departure from "traditional" SRL approaches that train on the hand-corrected syntactic trees associated with PropBank, we do not use any syntactic information from PropBank. Instead, we develop our models using automatically-generated syntax and named-entity (NE) labels, made available by the CoNLL shared task evaluation (Carreras and Màrquez, 2005). From the CoNLL data, our individual models based on full syntactic analysis use the trees generated by the Charniak parser. The partial-syntax model uses the chunk – i.e. basic syntactic phrase – labels and clause boundaries. All individual models make use of the provided NE labels.

Following the CoNLL-2005 setting we evaluated our system also on a fresh test set, derived from the Brown corpus. This second evaluation allows us to re-enforce our robustness claim.

3 Approach Overview

The proposed architecture, summarized in Figure 2, consists of two stages: a *candidate generation* phase and an *inference* stage.

In the candidate generation step, we merge the solutions of three individual SRL models into a unique pool of candidate arguments. The proposed models range from complete reliance on full parsing to using only partial syntactic information. The first two models, Model 1 and 2, are developed as sequential taggers (using the BIO tagging scheme) on a shared framework. The major difference between the two models is that Model 1 uses only partial syntactic information (basic phrases and clause boundaries), whereas Model 2 uses complete syntactic information. To maximize diversity, Model 3 implements a different strategy: it models only arguments that map into exactly one syntactic constituent. Section 4 details all three individual models.

The inference stage starts with *candidate filtering*,

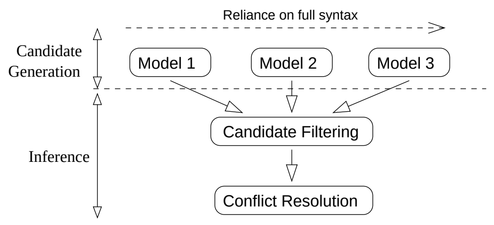


Figure 2: Architecture of the proposed system.

which reduces the number of candidate arguments in the pool using a single binary classifier. Using this classifier's confidence values and a number of domain-specific constraints, e.g. no two arguments can overlap, the *conflict resolution* component enforces the consistency of the final solution using a straightforward greedy strategy. The complete inference model is detailed in Section 5.

4 Individual SRL Models

Models 1 and 2. These models approach SRL as a sequential tagging task. In a pre-process step, the input syntactic structures are traversed in order to select a subset of constituents organized sequentially (i.e. non embedding). Model 1 makes use only of the partial tree defined by base chunks and clause boundaries, while Model 2 explores full parse trees.

Precisely, the sequential tokens are selected as follows. First, the input sentence is splitted into disjoint segments by considering the clause boundaries given by the syntactic structure. Second, for each segment, the set of top-most non-overlapping syntactic constituents completely falling inside the segment are selected as tokens. Note that this strategy provides a set of sequential tokens covering the complete sentence. Also, it is independent of the syntactic annotation explored, given it provides clause boundaries — see (Màrquez et al., 2005) for more details.

Due to this pre-processing stage, the upper-bound recall figures are 95.67% for Model 1 and 90.32% for Model 2 using the datasets defined in Section 6.

The nodes selected are labeled with B-I-O tags (depending if they are at the beginning, inside, or outside of a predicate argument) and they are converted into training examples by considering a rich set of features, mainly borrowed from state-of-the-art systems. These features codify properties from: (a) the argument constituent, (b) the target predicate,

Constituent *type* and *head*: extracted using common head-word rules. If the first element is a PP chunk, then the head of the first NP is extracted.
First and last words and POS tags of the constituent.
POS sequence: if it is less than 5 tags long. *2/3/4-grams* of the POS sequence.
Bag-of-words of nouns, adjectives, and adverbs.
TOP sequence: sequence of types of the top-most syntactic elements in the constituent (if it is less than 5 elements long). In the case of full parsing this corresponds to the right-hand side of the rule expanding the constituent node. *2/3/4-grams* of the TOP sequence.
Governing category as in (Gildea and Jurafsky, 2002).
NamedEnt, indicating if the constituent embeds or strictly matches a named entity along with its type.
TMP, indicating if the constituent embeds or strictly matches a temporal keyword (extracted from `AM-TMP` arguments of the training set).
Previous and following words and POS of the constituent.
The same features characterizing focus constituents are extracted for the *two previous and following tokens*, provided they are inside the clause boundaries of the codified region.

Table 1: Constituent structure features: Models 1/2

Predicate *form*, *lemma*, and *POS tag*.
Chunk type and *type of verb phrase* in which verb is included: single-word or multi-word.
The predicate *voice*. We currently distinguish five voice types: active, passive, copulative, infinitive, and progressive.
Binary flag indicating if the verb is a *start/end* of a clause.
Sub-categorization rule, i.e. the phrase structure rule that expands the predicate immediate parent.

Table 2: Predicate structure features: Models 1/2

and (c) the distance between the argument and predicate. The three feature sets are listed in Tables 1, 2, and 3, respectively.[1]

Regarding the learning algorithm, we used generalized AdaBoost with real-valued weak classifiers, which constructs an ensemble of decision trees of fixed depth (Schapire and Singer, 1999). We considered a one-vs-all decomposition into binary problems to address multi-class classification. AdaBoost binary classifiers are then used for *labeling* test sequences, from left to right, using a recurrent sliding window approach with information about the tag assigned to the preceding token. This tagging module enforces some basic constraints, e.g., BIO correct structure, arguments cannot overlap with clause nor chunk boundaries, discard `ARG0-5` arguments not present in PropBank frames for a certain verb, etc.

[1]Features extracted from partial parsing and Named Entities are common to Model 1 and 2, while features coming from full parse trees only apply to Model 2.

Relative position, *distance* in words and chunks, and *level of embedding* (in #clause-levels) with respect to the constituent.
Constituent path as described in (Gildea and Jurafsky, 2002) and all *3/4/5-grams* of path constituents beginning at the verb predicate or ending at the constituent.
Partial parsing path as described in (Carreras et al., 2004) and all *3/4/5-grams* of path elements beginning at the verb predicate or ending at the constituent.
Syntactic frame as described by Xue and Palmer (2004)

Table 3: Predicate–constituent features: Models 1/2

The *syntactic label* of the candidate constituent.
The constituent *head word*, *suffixes* of length 2, 3, and 4, *lemma*, and *POS tag*.
The constituent *content word*, *suffixes* of length 2, 3, and 4, *lemma*, *POS tag*, and *NE label*. Content words, which add informative lexicalized information different from the head word, were detected using the heuristics of (Surdeanu et al., 2003).
The *first and last constituent words* and their *POS tags*.
NE labels included in the candidate phrase.
Binary features to indicate the presence of *temporal cue words*, i.e. words that appear often in `AM-TMP` phrases in training.
For each TreeBank syntactic label we added a feature to indicate the *number of such labels* included in the candidate phrase.
The *sequence of syntactic labels* of the constituent immediate children.
The phrase *label*, *head word and POS tag* of the constituent parent, left sibling, and right sibling.

Table 4: Constituent structure features: Model 3

Model 3. The third individual SRL model makes the strong assumption that each predicate argument maps to one syntactic constituent. For example, in Figure 1 `ARG0` maps to a noun phrase, `ARGM-LOC` maps to a prepositional phrase etcetera. This assumption holds well on hand-corrected parse trees and simplifies significantly the SRL process because only one syntactic constituent has to be correctly classified in order to recognize one semantic argument. On the other hand, this approach is limited when using automatically-generated syntactic trees. For example, only slightly over 91% of the arguments can be mapped to one of the syntactic constituents produced by the Charniak parser.

Using a bottom-up approach, Model 3 maps each argument to the first syntactic constituent that has the exact same boundaries and then climbs as high as possible in the tree across unary production chains. We currently ignore all arguments that do not map to a single syntactic constituent.

The predicate *word* and *lemma*.
The predicate *voice*. Same definition as Models 1 and 2.
A binary feature to indicate if the predicate is *frequent* (i.e., it appears more than twice in the training data) or not.
Sub-categorization rule. Same def. as Models 1 and 2.

Table 5: Predicate structure features: Model 3

The *path* in the syntactic tree between the argument phrase and the predicate as a chain of syntactic labels along with the traversal direction (up or down).
The *length* of the above syntactic path.
The *number of clauses* (`S*` phrases) in the path.
The *number of verb phrases* (`VP`) in the path.
The *subsumption count*, i.e. the difference between the depths in the syntactic tree of the argument and predicate constituents. This value is 0 if the two phrases share the same parent.
The *governing category*, which indicates if `NP` arguments are dominated by a sentence (typical for subjects) or a verb phrase (typical for objects).
We *generalize* syntactic paths with more than 3 elements using two templates: (a) $\texttt{Arg} \uparrow \texttt{Ancestor} \downarrow \texttt{N}_i \downarrow \texttt{Pred}$, where `Arg` is the argument label, `Pred` is the predicate label, `Ancestor` is the label of the common ancestor, and $\texttt{N}_i$ is instantiated with all the labels between `Pred` and `Ancestor` in the full path; and (b) $\texttt{Arg} \uparrow \texttt{N}_i \uparrow \texttt{Ancestor} \downarrow \texttt{Pred}$, where $\texttt{N}_i$ is instantiated with all the labels between `Arg` and `Ancestor` in the full path.
The *surface distance* between the predicate and the argument phrases encoded as: the number of tokens, verb terminals (`VB*`), commas, and coordinations (`CC`) between the argument and predicate phrases, and a binary feature to indicate if the two constituents are adjacent.
A binary feature to indicate if the argument *starts with a predicate particle*, i.e. a token seen with the `RP*` POS tag and directly attached to the predicate in training.

Table 6: Predicate–constituent features: Model 3

Once the mapping process completes, Model 3 extracts a rich set of lexical, syntactic, and semantic features. Tables 4, 5, and 6 present these features organized in the same three categories as the previous Models 1 and 2 — see (Surdeanu and Turmo, 2005) for more details.

Similarly with Models 1 and 2, Model 3 trains one-vs-all classifiers using AdaBoost for the most common argument labels. To reduce the sample space, Model 3 selects training examples (both positive and negative) only from: (a) the first clause that includes the predicate, or (b) from phrases that appear to the left of the predicate in the sentence. More than 98% of the argument constituents fall into one of these classes.

At prediction time the classifiers are combined using a simple greedy technique that iteratively assigns to each predicate the argument classified with the highest confidence. For each predicate we consider as candidates all AM attributes, but only numbered attributes indicated in the corresponding PropBank frame. Additionally, this greedy strategy enforces a limited number of domain knowledge constraints in the generated solution: (a) arguments can not overlap in any form, and (b) no duplicate arguments are allowed for ARG0-5.

5 The Inference Model

The most important component of our inference model is candidate filtering, which decides if a candidate argument should be maintained in the global solution or not. Candidate filtering is implemented as a single binary classifier that uses only features extracted from the solutions provided by the individual systems. For robustness, we do not use any features that are specific to any of the individual models, nor the confidence value of their classifiers.

Table 7 lists the features extracted from each candidate argument by the filtering classifier. For simplicity we have focused only on attributes that: (a) are readily available in the solutions proposed by the individual classifiers, and (b) allow the gathering of simple and robust statistics. For example, the filtering classifier might learn that a candidate is to be trusted if: (a) two individual systems proposed it, (b) if its label is ARG2 and it was generated by Model 1, or (c) if it was proposed by Model 2 within a certain argument sequence.

The candidate arguments that pass the filtering stage are incorporated in the global solution by the conflict resolution module, which enforces several domain specific constraints. We have currently implemented two constraints: (a) arguments can not overlap or embed other arguments; and (b) no duplicate arguments are allowed for the numbered arguments ARG0-5. Theoretically, the set of constraints can be extended with any other rules, but in our particular case, we know that some constraints, e.g. providing only arguments indicated in the corresponding PropBank frame, are already guaranteed by the individual models. Conflicts are solved with a straightforward greedy strategy: the pool of candidate arguments is inspected in descending order of the confidence values assigned by the filtering classifier, and candidates are appended to the global solution only if they do not violate any of the domain constraints with the arguments already selected. Our inference system currently has a sequential architecture, i.e. no feedback is sent from the conflict resolution module to candidate filtering.

The *label* of the candidate argument.
The *number of systems* that generated an argument with this label and span.
The *unique ids*, e.g. M1 and M2, of all the systems that generated an argument with this label and span.
The *argument sequence* for this predicate for all the systems that generated an argument with this label and span. For example, the argument sequence for the proposition illustrated in Figure 1 is: ARG0 - ARGM-TMP - P - ARG1 - ARGM-LOC.
The *number* and *unique ids* of all the systems that generated an argument with the *same span but different label*.
The *number* and *unique ids* of all the systems that generated an argument *included* in the current argument.
The *number* and *unique ids* of all the systems that generated an argument that *contains* the current argument.
The *number* and *unique ids* of all the systems that generated an argument that *overlaps* the current argument.

Table 7: Features used by the candidate filtering classifier.

6 Experimental Results

We trained the individual models using the complete CoNLL-2005 training set (PropBank/TreeBank sections 2 to 21). All models were developed using AdaBoost with decision trees of depth 4 (i.e. each branch may represent a conjunction of at most 4 basic features). Each classification model was trained for up to 2,000 rounds.

We applied some simplifications to keep training times and memory requirements inside admissible bounds: (a) we have limited the number of negative examples in Model 3 to the first 500,000; (b) we have trained only the most frequent argument labels: top 41 for Model 1, top 35 for Model 2, and top 24 for Model 3; and (c) we discarded all features occurring less than 15 times in the training set.

The models were tuned on a separate development partition (TreeBank section 24) and evaluated on two corpora: (a) TreeBank section 23, which consists of Wall Street Journal (WSJ) documents, and (b) on three sections of the Brown corpus, semantically annotated by the PropBank team for the CoNLL 2005 shared task evaluation. Table 8 sum-

WSJ	PProps	Precision	Recall	$F_{\beta=1}$
Model 1	48.45%	78.76%	72.44%	75.47 ± 0.8
Model 2	**52.04%**	79.65%	**74.92%**	**77.21** ± 0.8
Model 3	45.28%	**80.32%**	72.95%	76.46 ± 0.6
Brown	PProps	Precision	Recall	$F_{\beta=1}$
Model 1	30.85%	67.72%	58.29%	62.65 ± 2.1
Model 2	**36.44%**	71.82%	**64.03%**	**67.70** ± 1.9
Model 3	29.48%	**72.41%**	59.67%	65.42 ± 2.1

Table 8: Overall results of the individual models on the WSJ and Brown test sets.

marizes the results of the three models on the WSJ and Brown corpora. In that table we include the percentage of perfect propositions detected by each model ("PProps"), i.e. predicates recognized with all their arguments, the overall precision, recall, and $F_{\beta=1}$ measure[2].

The results summarized in Table 8 indicate that all individual systems have a solid performance. Although none of them would rank in the top 3 in this year's CoNLL evaluation (Carreras and Màrquez, 2005), their performance is comparable to the best individual systems presented at that evaluation exercise[3]. As expected, the models based on full parsing (2 and 3) perform better than the model based on partial syntax. But, interestingly, the difference is not large (e.g., less than 2 points in $F_{\beta=1}$ in the WSJ corpus), evincing that having base syntactic chunks and clause boundaries is enough to obtain a competitive performance with a simple system.

Consistently with other systems evaluated on the Brown corpus, all our models experience a severe performance drop in this corpus, due to the lower performance of the linguistic processors.

6.1 Performance of Combination Systems

We have trained the candidate filtering binary classifier on one third of the training partition. Its training data was generated using individual models trained on the other two thirds of the training partition. The classifier was developed using Support Vector Machines (SVM) with a polynomial kernel of degree 2. We trained combined models for all 4 possible combinations of our 3 individual models.

[2]The signifi cance intervals for the F_1 measure have been obtained using bootstrap resampling (Noreen, 1989). F_1 rates outside of these intervals are assumed to be signifi cantly different from the related F_1 rate ($p < 0.05$).

[3]The best performing SRL systems at CoNLL were a combination of several subsystems. See section 7 for details.

Table 9 summarizes the performance of the combined systems on the WSJ and Brown corpora.[4] The combined systems are compared against a baseline combination system, which merges *all* the arguments generated by the individual systems. For conflict resolution, the baseline uses the greedy strategy introduced in Section 5, but using as argument ordering criterion a radix sort that orders the candidate arguments in descending order of: number of models that agreed on this argument; argument length in tokens; and performance of the individual system[5].

Table 9 indicates that our combination strategy is always successful: the results of all combined systems improve upon their individual models and they are better the baseline when using the same number of individual models. As expected, the highest scoring combined system includes all three individual models. Its $F_{\beta=1}$ measure is 2.35 points higher than the best individual model (Model 2) in the WSJ test set and 1.30 points higher in the Brown test set. Somewhat surprisingly, the highest percentage of perfect propositions is not obtained by the overall best combination, but by the system that combines the two models based on full parsing (Models 2 and 3). This happens because Model 1 is the weakest performing of the bunch, hence its arguments, while providing useful information to the filtering classifier, decrease the number of perfect propositions when selected.

We consider these results encouraging given the simplicity of our inference model and the limited amount of training data used to train the candidate filtering classifier. Additionally, they compare favorably with respect to the best performing systems at CoNLL-2005 shared task (see Section 7).

6.2 Upper Limit of the Combination Strategy

To explore the potential of our approach we have constructed a hypothetical system where our candidate filtering module is replaced with a perfect classifier that selects only correct arguments and discards all others. Table 10 lists the results obtained on the WSJ and Brown corpora by this hypothetical system using all three individual models.

[4]For conciseness, in Table 9 we introduced the notation M1+2+3 to indicate the combination of Models 1, 2, and 3

[5]This combination produced the highest-scoring baseline model.

WSJ	PProps	Prec.	Recall	$F_{\beta=1}$
M1+2	51.30%	81.30%	74.13%	77.55 ± 0.7
M1+3	47.26%	81.21%	73.36%	77.08 ± 0.8
M2+3	**52.65**%	81.55%	75.30%	78.30 ± 0.7
M1+2+3	51.64%	**84.89**%	74.87%	**79.56** ± 0.7
baseline	51.09%	77.29%	**78.67**%	77.98 ± 0.7
Brown	**PProps**	**Prec.**	**Recall**	$F_{\beta=1}$
M1+2	35.95%	73.70%	62.93%	67.89 ± 2.0
M1+3	28.98%	72.83%	58.84%	65.09 ± 2.2
M2+3	**37.06**%	73.89%	63.30%	68.18 ± 2.2
M1+2+3	34.20%	**78.66**%	61.46%	**69.00** ± 2.1
baseline	33.58%	67.66%	**66.01**%	66.82 ± 1.8

Table 9: Overall results of the combination models on the WSJ and Brown test sets.

	Perfect props	Precision	Recall	$F_{\beta=1}$
WSJ	70.76%	99.12%	85.22%	91.64
Brown	51.87%	99.63%	74.32%	85.14

Table 10: Performance upper limit on the test sets.

Table 10 indicates that the upper limit of proposed approach is relatively high: the $F_{\beta=1}$ of this hypothetical system is over 12 points higher than our best combined system in the WSJ test set, and over 16 points higher in the Brown corpus. These results indicate that the potential of our combination strategy is high, especially when compared with re-ranking strategies, which are limited to the performance of the best *complete* solution in the candidate pool. By allowing the re-combination of arguments from the individual candidate solutions we raise this threshold significantly. Table 11 lists the contribution of the individual models to this upper limit on the WSJ corpus. For conciseness, we list only the "core" numbered arguments. "$\cap$ of 3" indicates the percentage of correct arguments where all 3 models agreed, "$\cap$ of 2" indicates the percentage of correct arguments where any 2 models agreed, and the other columns indicate the percentage of correct arguments detected by a single model. Table 11 indicates that, as expected, two or more individual models agreed on a large percentage of the correct arguments. Nevertheless, a significant number of correct arguments, e.g. over 22% of ARG3, come from a *single* individual system. This proves that, in order to achieve maximum performance, one has to look beyond simple voting strategies that favor arguments with high agreement between individual systems.

	$\cap$ of 3	$\cap$ of 2	M1	M2	M3
ARG0	80.45%	12.10%	3.47%	2.14%	1.84%
ARG1	69.82%	17.83%	7.45%	2.77%	2.13%
ARG2	56.04%	22.32%	12.20%	4.95%	4.49%
ARG3	56.03%	21.55%	12.93%	5.17%	4.31%
ARG4	65.85%	20.73%	6.10%	2.44%	4.88%

Table 11: Contribution of the individual systems to the upper limit, for ARG0–ARG4 in the WSJ test set.

	WSJ		Brown	
	PProps	$F_{\beta=1}$	PProps	$F_{\beta=1}$
koomen	53.79%	**79.44** ± 0.8	32.34%	**67.75** ± 1.8
haghighi	**56.52**%	78.45 ± 0.8	**37.06**%	67.71 ± 2.0
pradhan	50.14%	77.37 ± 0.7	36.44%	67.07 ± 2.0

Table 12: Results of the best combined systems at CoNLL-2005.

7 Related Work

The best performing systems at the CoNLL-2005 shared task included a combination of different base subsystems to increase robustness and to gain coverage and independence from parse errors. Therefore, they are closely related to the work of this paper. Table 12 summarizes their results under exactly the same experimental setting.

Koomen et al. (2005) used a 2 layer architecture similar to ours. The pool of candidates is generated by running a full syntax SRL system on alternative input information (Collins parsing, and 5-best trees from Charniak's parser). The combination of candidates is performed in an elegant global inference procedure as constraint satisfaction, which, formulated as Integer Linear Programming, can be solved efficiently. Interestingly, the generalized inference layer allows to include in the objective function, jointly with the candidate argument scores, a number of linguistically-motivated constraints to obtain a coherent solution. Differing from the strategy presented in this paper, their inference layer does not include learning. Also, they require confidence values from individual classifiers. This is the best performing system at CoNLL-2005.

Haghighi et al. (2005) implemented a double re-ranking model on top of the base SRL models to select the most probable solution among a set of candidates. The re-ranking is performed, first, on a set of n-best solutions obtained by the base system run on a single parse tree, and, then, on the set of best-candidates coming from the n-best parse trees. The re-ranking approach allows to define global complex features applying to complete candidate solutions to train the rankers. The main drawback, compared to our approach, is that re-ranking does not permit to combine different solutions since it is forced to select a complete candidate solution. This fact implies that the performance upper limit strongly depends on the ability of the base model to generate the complete correct solution in the set of n-best candidates.

Finally, Pradhan et al. (2005b) followed a stacking approach by learning two individual systems based on full syntax, whose outputs are used to generate features to feed the training stage of a final chunk-by-chunk SRL system. Although the fine granularity of the chunking-based system allows to recover from parsing errors, we find this combination scheme quite ad-hoc because it forces to break argument candidates into chunks in the last stage.

8 Conclusions

This paper introduces a novel, robust combination strategy for semantic role labeling. Our approach is separated in two stages: a candidate generation phase, which combines the solutions generated by several individual models into a pool of candidate arguments, followed by a simple inference model that filters the candidate arguments using a single binary classifier and then enforces an arbitrary number of domain-specific constraints.

The proposed approach has several advantages. First, because it combines the solutions provided by the individual models, the inference model can recover from errors produced in the generation phase. Second, due to the diversity of the individual models employed, the candidate pool contains a high percentage of the correct arguments. And lastly, our approach is flexible and robust: it can incorporate any SRL model in the candidate generation stage because it does not require that the individual SRL models provide any information, e.g. classification confidence values, other than an argument solution.

Our results are better than the state of the art using automatically-generated syntactic information. These results are encouraging considering the simplicity of the proposed approach.

Acknowledgments

This research has been partially supported by the European Commission (CHIL project, IP-506909). Mihai Surdeanu is a research fellow within the Ramón y Cajal program of the Spanish Ministry of Education and Science.

References

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL 2004*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

X. Carreras, L. Màrquez, and G. Chrupała. 2004. Hierarchical recognition of propositional arguments with perceptrons. In *Proceedings of CoNLL 2004 shared task*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

A. Haghighi, K. Toutanova, and C. Manning. 2005. A joint model for semantic role labeling. In *Proceedings of CoNLL-2005 shared task*.

P. Kingsbury, M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the Penn Treebank. In *Proceedings of the Human Language Technology Conference*.

P. Koomen, V. Punyakanok, D. Roth, and W. Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005 shared task.*

L. Màrquez, P. Comas, J. Giménez, and N. Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL-2005 shared task*.

S. Narayanan and S. Harabagiu. 2004. Question answering based on semantic structures. In *International Conference on Computational Linguistics (COLING 2004)*.

E. W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005a. Support vector learning for semantic argument classifi cation. *Machine Learning, to appear*.

S. Pradhan, K. Hacioglu, W. Ward, J. H. Martin, and D. Jurafsky. 2005b. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL-2005.*

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confi dence-rated predictions. *Machine Learning*, 37(3).

M. Surdeanu and J. Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of CoNLL-2005 shared task.*

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.

A Methodology for Extrinsically Evaluating Information Extraction Performance

Michael Crystal, Alex Baron, Katherine Godfrey, Linnea Micciulla, Yvette Tenney, and Ralph Weischedel
BBN Technologies
10 Moulton St.
Cambridge, MA 02138-1119
mcrystal@bbn.com

Abstract

This paper reports a preliminary study addressing two challenges in measuring the effectiveness of information extraction (IE) technology:

- Developing a methodology for extrinsic evaluation of IE; and,
- Estimating the impact of improving IE technology on the ability to perform an application task.

The methodology described can be employed for further controlled experiments regarding information extraction.

1 Introduction

Intrinsic evaluations of information extraction (IE) have a history dating back to the Third Message Understanding Conference[1] (MUC-3) and continuing today in the Automatic Content Extraction (ACE) evaluations.[2] *Extrinsic* evaluations of IE, measuring the utility of IE in a task, are lacking and needed (Jones, 2005).

In this paper, we investigate an extrinsic evaluation of IE where the task is question answering (QA) given extracted information. In addition, we propose a novel method for exploring hypothetical performance questions, e.g., if IE accuracy were x% closer to human accuracy, how would speed and accuracy in a task, e.g., QA, improve?

We plot QA accuracy and time-to-complete given eight extracted data accuracy levels ranging from the output of SERIF, BBN's state-of-the-art IE system, to manually extracted data.

2 Methodology

Figure 1 gives an overview of the methodology. The left portion of the figure shows source documents provided both to a system and a human to produce two extraction databases, one corresponding to SERIF's automated performance and one corresponding to double-annotated, human accuracy. By merging portions of those two sources in varying degrees ("blends"), one can derive several extracted databases ranging from machine quality, through varying percentages of improved performance, up to human accuracy. This method of blending databases provides a means of answering hypothetical questions, i.e., what if the state-of-the-art were x% closer to human accuracy, with a single set of answer keys.

A person using a given extraction database performs a task, in our case, QA. The measures of effectiveness in our study were time to complete the task and percent of questions answered correctly. An extrinsic measure of the value of improved IE technology performance is realized by rotating users through different extraction databases and questions sets.

In our preliminary study, databases of fully automated IE and manual annotation (the gold standard) were populated with entities, relationships, and co-reference links from 946 documents. The two initial databases representing machine extraction and human extraction respectively were then blended to produce a continuum of database qualities from machine to

[1] For more information on the MUC conferences, see http://www.itl.nist.gov/iad/894.02/related_projects/muc/.
[2] For an overview of ACE evaluations see http://www.itl.nist.gov/iad/894.01/tests/ace/.

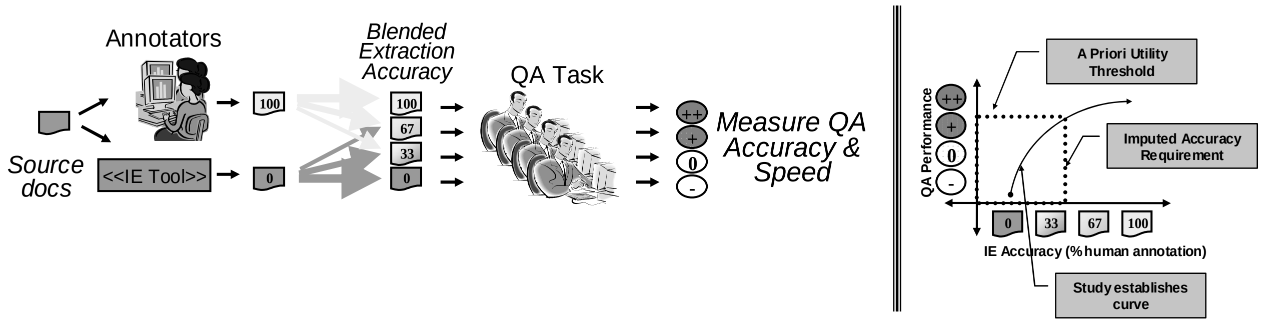

Annotators
Blended Extraction Accuracy
QA Task
Source docs
<<IE Tool>>
100
67
33
0
++
+
0
-
Measure QA Accuracy & Speed
A Priori Utility Threshold
Imputed Accuracy Requirement
QA Performance
IE Accuracy (% human annotation)
Study establishes curve

Figure 1: Study Overview

human performance. ACE Value Scores[3] were measured for each database. Pilot studies were conducted to develop questions for a QA task. Each participant answered four sets of questions, each with a different extraction database representing a different level of IE accuracy. An answer capture tool recorded the time to answer each question and additional data to confirm that the participant followed the study protocol. The answers were then evaluated for accuracy and the relationship between QA performance and IE quality was established.

Each experiment used four databases. The first experiment used databases spanning the range from solely machine extraction to solely human extraction. Based on the results of this experiment, two further experiments focused on smaller ranges in database quality to study the relationship between IE and QA performance.

2.1 Source Document Selection, Annotation, and Extraction

Source documents were selected based on the availability of manual annotation. We identified 946 broadcast news and newswire articles from recent ACE efforts, all annotated by the LDC according to the ACE guidelines for the relevant year (2002, 2003, 2004). Entities, relations, and within-document co-reference were marked. Inter-document co-reference annotation was added by BBN. The 946 news articles comprised 363 articles (187,720 words) from newswire and 583 (122,216 words) from broadcast news. With some corrections to deal with errors and changes in guidelines, the annotations were loaded as the human (DB-quality 100) database.

SERIF, BBN's automatic IE system based on its predecessor, SIFT (Miller, 2000), was run on the 946 ACE documents to create the machine (DB-quality 0) database. SERIF is a statistically trained software system that automatically performs entity, co-reference, and relationship information extraction.

Intermediate IE performance was simulated by blending the human and automatically generated databases in various degrees using an interpolation algorithm developed specifically for this study. To create a blended database, DB-quality *n*, all of the entities, relationships, and co-reference links common to the human and automatically generated databases are copied into a new one. Then, *n*% of the entity mentions in the human database (100), but not in the automatic IE system output (0), are copied; and, (100 – *n*)% of the entity mentions in the automatically generated database, but not in the human database, are copied. Next, the relationships for which both of the constituent entity mentions have been copied are also copied to the blended database. Finally, co-reference links and entities for the already copied entity mentions are copied into the blended database.

For the first experiment, two intermediate extraction databases were created: DB-qualities 33 and 67. For the second experiment, two additional databases were created: 16.5 and 50. The first intermediate databases were both created using the 0 and 100 databases as seeds. The 16.5 database was created by mixing the 0 and the 33 databases in a 50% blend. The 50 database was created by doing the same with the 33 and 67 databases. For Experiment 3, 41 and 58 databases were created by mixing the 33 and 50, and 50 and 67 databases respectively.

[3] The 2004 ACE evaluation plan, available at http://www.nist.gov/speech/tests/ace/ace04/doc/ace04-evalplan-v7.pdf, contains a full description of the scoring metric used in the evaluation. Entity type weights were 1 and the level weights were NAM=1.0, NOM=0.5, and PRO=0.1.

	DB Blend															
	0 (Machine)		16.5		33		41		50		58		67		100 (Human)	
	Ent	Rel	Ent	Rel	Ent	Rel	Ent	Rel	Ent	Rel	Ent	Rel	Ent	Rel	Ent	Rel
Recall	64	33	70	40	74	45	76	48	79	54	82	58	86	65	100	100
Pre.	74	50	77	62	79	67	80	70	83	75	85	78	89	82	100	100
Value	60	29	67	37	71	42	73	45	77	51	80	56	84	63	100	100

Table 1: Precision, Recall and Value Scores for Entities and Relations for each DB Blend

	0 (Machine)	16.5	33	41	50	58	67	100 (Human)
Entities	17,117	18,269	18,942	19,398	19,594	19,589	19,440	18,687
Relations	6,684	6,675	6,905	7,091	7,435	7,808	8,406	11,032
Descriptions	18,666	18,817	19,135	19,350	19,475	19,639	19,752	20,376

Table 2: Entity, Relation and Description Counts for each DB Blend

To validate the interpolation algorithm and blending procedure, we applied NIST's 2004 ACE Scorer to the eight extraction databases. Polynomial approximations were fitted against both the entity and relation extraction curves. Entity performance was found to vary linearly with DB blend ($R^2 = .9853$) and relation performance was found to vary with the square of DB blend ($R^2 = .9961$). Table 1 shows the scores for each blend, and Table 2 shows the counts of entities, relationships, and descriptions.

2.2 Question Answering Task

Extraction effectiveness was measured by how well a person could answer questions given a database of facts, entities, and documents. Participants answered four sets of questions using four databases. They accessed the database using BBN's FactBrowser (Miller, 2001) and recorded their answers and source citations in a separate tool developed for this study, AnswerPad.

Each database represented a different database quality. In some databases, facts were missing, or incorrect facts were recorded. Consequently, answers were more accessible in some databases than in others, and participants had to vary their question answering strategy depending on the database.

Participants were given five minutes to answer each question. To ensure that they had actually located the answer rather than relied on world knowledge, they were required to provide source citations for every answer. The instructions emphasized that the investigation was a test of the system, and not of their world knowledge or web search skills. Compliance with these instructions was high. Users resorted to knowledge-based proper noun searches only one percent of the time. In addition, keyword search was disabled to force participants to rely on the database features.

2.3 Participants

Study participants were recruited through local web lists and at local colleges and universities. Participants were restricted to college students and recent graduates with PC (not Mac) experience, without reading disabilities, for whom English was their native language. No other screening was necessary because the design called for each participant to serve as his or her own control, and because opportunities to use world knowledge in answering the questions were minimized through the interface and procedures.

During the first two months of the study 23 participants were used to help develop questions, participant criteria, and the overall test procedure. Then, experiments were conducted comparing the 0, 33, 67, and 100 database blends (Experiment 1, 20 subjects); the 0, 16.5, 33, and 50 database blends (Experiment 2, 20 subjects), and the 33, 41, 50, and 58 database blends (Experiment 3, 24 subjects).

2.4 Question Selection and Validation

Questions were developed over two months of pilot studies. The goal was to find a set of questions that would be differentially supported by the 0, 33, 67, and 100 databases. We explored both "random" and "engineered" approaches. The random approach called for creating questions using only the documents, without reference to the kind of information extracted. Using a list of keywords, one person generated 86 questions involving relationships and entities pertaining to politics and the military by scanning the 946 ACE documents to find references to each keyword and devising questions based on the information she found.

The alternative, engineered approach involved eliminating questions that were not supported by the types of information extracted by SERIF, and generating additional questions to fit the desired pattern of increasing support with increased human annotation. This approach ensured that the question sets reflected the structural differences that are assumed to exist in the database, and produced psychophysical data that link degree of QA support to human performance parameters. The IE results from four of the databases (0, 33, 67 and 100) were used to develop questions that received differential support from the different quality databases. For example, such a question could be answered using the automatically extracted results, but might be more straightforwardly answered given human annotation.

Sixty-four questions, plus an additional ten practice questions, were created using the engineering approach. Additional criteria that were followed in creating the question sets were: 1) Questions had to contain at least one reasonable entry hook into all four databases, e.g., the terms U.S. and America were considered too broad to be reasonable; and, 2) For ease of scoring, list-type questions had to specify the number of answers required. Alternative criteria were considered but rejected because they correlated with the aforementioned set. The following are examples of engineered questions.

- Identify eight current or former U.S. State Department workers.
- In what two West Bank towns does Fatah have an office?
- Name two countries where Osama bin Laden has been.
- Were Lebanese women allowed to vote in municipal elections between two Shiite groups in the year 1998?

Two question lists, one with 86 questions generated by the random procedure and one with 64 questions generated by the engineered procedure, were analyzed with respect to the degree of support afforded by each of the four databases as viewed through FactBrowser. Four *a priori* criteria were established to assess degree of support – or its opposite, the degree of expected difficulty – for each question in each of the four databases. Ranked from easiest to hardest, they are listed in Table 3.

The question can be answered…
1. Directly with fact or description (answer is highlighted in FactBrowser citation)
2. Indirectly with fact or description (answer is not highlighted)
3. With name mentioned in question (long list of mentions without context)
4. Via database crawling

Table 3: *A Priori* Question Difficulty Characteristics, listed from easiest to hardest

Table 4 shows the question difficulty levels for both question types, for each of four databases. Analysis of the engineered set was done on all 64 questions. Analysis for randomly generated questions was done on a random sample of 44 of the 86 questions. Fifteen questions did not meet the question criteria, leaving 29.

The randomly generated questions showed a statistically significant, but small, variation in expected difficulty, in part due to the number of unanswerable questions. While the questions were made up with respect to information found in the documents, the process did not consider the types of extracted entities and relations. This problem might have been mitigated by limiting the search to questions involving entities and relations that were part of the extraction task.

By contrast, the engineered question set showed a highly significant decrease in expected difficulty as the percentage of human annotation in the database increased ($P < 0.0001$ for chi-square analysis). This result is not surprising, given that the questions were constructed with reference to the list of entities in the four databases. The analysis confirms that the experimental manipulation of different degrees of support provided by the four databases was achieved for this question set.

Random Question Generation

Difficulty Level (easiest to hardest)	0% Human	33% Human	67% Human	100% Human
1 Fact-Highlight	7	10	13	15
2 Fact-Indirect	14	10	8	10
3 Mention	3	5	2	1
4 Web Crawl	5	4	6	3
Total	29	29	29	29

Engineered Question Generation

Difficulty Level (from easiest to hardest)	0% Human	33% Human	67 Human	100% Human
1 Fact-Highlight	16	25	35	49
2 Fact-Indirect	23	20	18	14
3 Mention	7	14	11	1
4 Web Crawl	18	5	0	0
Total	64	64	64	64

Table 4: Anticipated Difficulty of Questions as a Function of Database Quality

Preliminary human testing with both question sets suggested that the *a priori* difficulty indicators predict human question answering performance. Experiments with the randomly generated questions, therefore, were unlikely to reveal much about the databases or about human question answering performance. On the other hand, an examination of how different levels of database quality affect human performance, in a psychophysical experiment where structure is varied systematically, promised to address the question of how much support is needed for good performance.

Based on the question difficulties, and pilot study timing and performance results, the 64 questions were grouped into four, 16-question balanced sets.

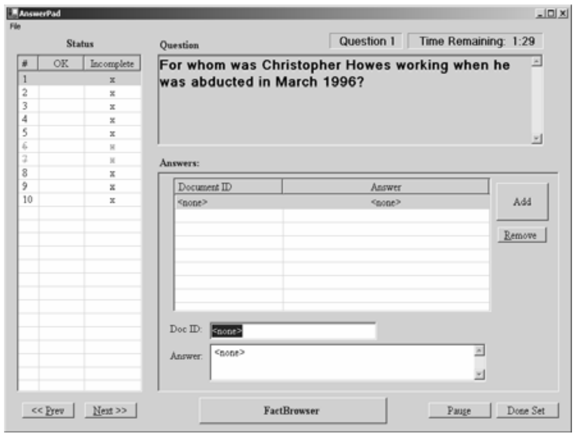
2.5 Procedure

Participants were tested individually at our site, in sessions lasting roughly four hours. Training prior to the test lasted for approximately a half hour. Training consisted of a walk-through of the interface features followed by guided practice with sample questions. The test consisted of four question sets, each with a different database. Participants were informed that they would be using a different database for each question set and that some might be easier to use than others.

Questions were automatically presented and responses were captured in AnswerPad, a software tool designed for the study. AnswerPad is shown in Figure 2.

Key features of the tool include:

- Limiting view to current question set – disallowing participants to view previous question sets
- Automatically connecting to correct db
- Logging time spent on each question
- Enforcing five-minute limit per question
- Enforcing requirement that all answers include a citation



For whom was Christopher Howes working when he was abducted in March 1996?

#	OK	Incomplete
1		x
2		x
3		x
4		x
5		x
6		x
7		x
8		x
9		x
10		x

Document ID	Answer
<none>	<none>

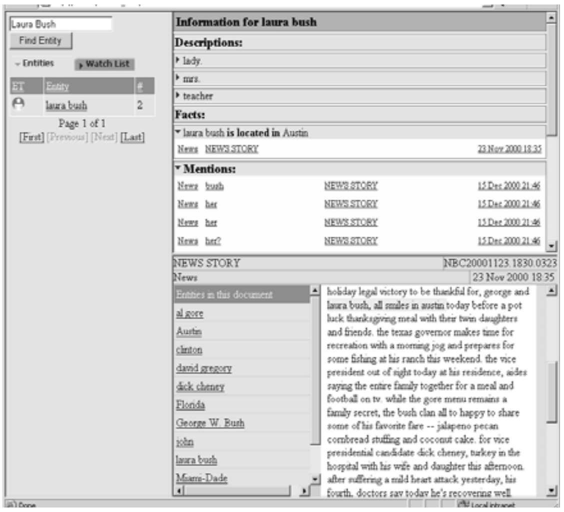
Doc ID: <none>

Answer: <none>

Figure 2: AnswerPad Question Presentation and Answer Capture Interface

Participants were given written documentation as part of their training. The participants were instructed to cut-and-paste question answers and document citations from source documents into AnswerPad.

Extracted facts and entities, and source documents were accessed through FactBrowser. FactBrowser, shown in Figure 3, is web-browser based and is invoked via a button in AnswerPad. FactBrowser allows one to enter a string, which is matched against the database of entity mentions. The list of entities that have at least one mention partially matching the string are returned (e.g., “Laura Bush”) along with an icon indicating the type of the entity and the number of documents in which the entity appears. Clicking on the entity in the left panel causes the top right panel to display all of the descriptions, facts, and mentions for the entity. Selecting one of these displays citations in which the description, fact, or mention occurs. Clicking on the citation opens up a document view in the lower right corner of the screen and highlights the extracted information in the text. When a document is displayed, all of the entities detected in the document are listed down the left side of the document viewer.



ET	Entity	#
	laura bush	2

Information for laura bush

Descriptions:
- lady.
- mrs.
- teacher

Facts:
- laura bush **is located in** Austin

News NEWS STORY 23 Nov 2000 18:35

Mentions:

News	bush	NEWS STORY	15 Dec 2000 21:46
News	her	NEWS STORY	15 Dec 2000 21:46
News	her	NEWS STORY	15 Dec 2000 21:46
News	her?	NEWS STORY	15 Dec 2000 21:46

NEWS STORY NBC20001123.1830.0323
News 23 Nov 2000 18:35

Entities in this document: al gore, Austin, clinton, david gregory, dick cheney, Florida, George W. Bush, john, laura bush, Miami-Dade

holiday legal victory to be thankful for, george and laura bush, all smiles in austin today before a pot luck thanksgiving meal with their twin daughters and friends. the texas governor makes time for recreation with a morning jog and prepares for some fishing at his ranch this weekend. the vice president out of sight today at his residence, aides saying the entire family together for a meal and football on tv. while the gore menu remains a family secret, the bush clan all to happy to share some of his favorite fare -- jalapeno pecan cornbread stuffing and coconut cake. for vice presidential candidate dick cheney, turkey in the hospital with his wife and daughter this afternoon. after suffering a mild heart attack yesterday, his fourth. doctors say today he's recovering well.

Figure **3**: Browsing Tool Interface

The browsing tool was instrumented to record command invocations so that the path a participant took to answer a question could be recreated, and the participant’s adherence to protocol could be verified. Furthermore, the find function (Ctrl-F) was disabled to prevent users from performing *ad hoc* searches of the documents instead of using the extracted data.

The order of question sets and the order of database conditions were counterbalanced across participants, so that, for every four participants, every question set and database appeared once in every ordinal position, and every question set was paired once with every database. This avoided carryover effects from question order.

2.6 Data Collected

Based on the initial results from Experiment 1, a 70% target effectiveness threshold was identified to occur between the 33 and 67 database blends. To refine and verify this finding, Experiment 2 examined the 0, 16.5, 33, and 50 database blends. Experiment 3 examined the 33, 41, 50, and 58 database blends.

AnswerPad collected participant-provided answers to questions and the corresponding citations. In addition, AnswerPad recorded the time spent answering the questions. A limit of five minutes was imposed based on pilot study results. The browsing tool logged commands invoked while the user searched the fact-base for question answers. Questions were manually scored based on the answers in the provided corpus. No partial credit was given. The maximum score, for each database condition, was 16, for a total maximum score of 64.

3 Results

Figure 4 shows the question answer scores and times for each of the three individual experiments, and for Experiments 1 and 2 combined. Database quality affects both task speed (downward-sloping line) and task accuracy (upward-sloping line) in the expected direction. A logistic fit, as for a binary-response curve, was used to fit the relationship between blend percentage and accuracy in each experiment. The logistic fit Goodman-Theil quasi-R^2 was .9973 for Experiment 1, .9594 for Experiment 2, .8936 for Experiment 3, and .9959 for Experiments 1 and 2 combined.

For the target accuracy of 70%, the 95% confidence interval for the required blend is (35,56) around a predicted 46% blend for Experiment 1, and (41,56) around a predicted 49% for Experiments 1 and 2 combined.

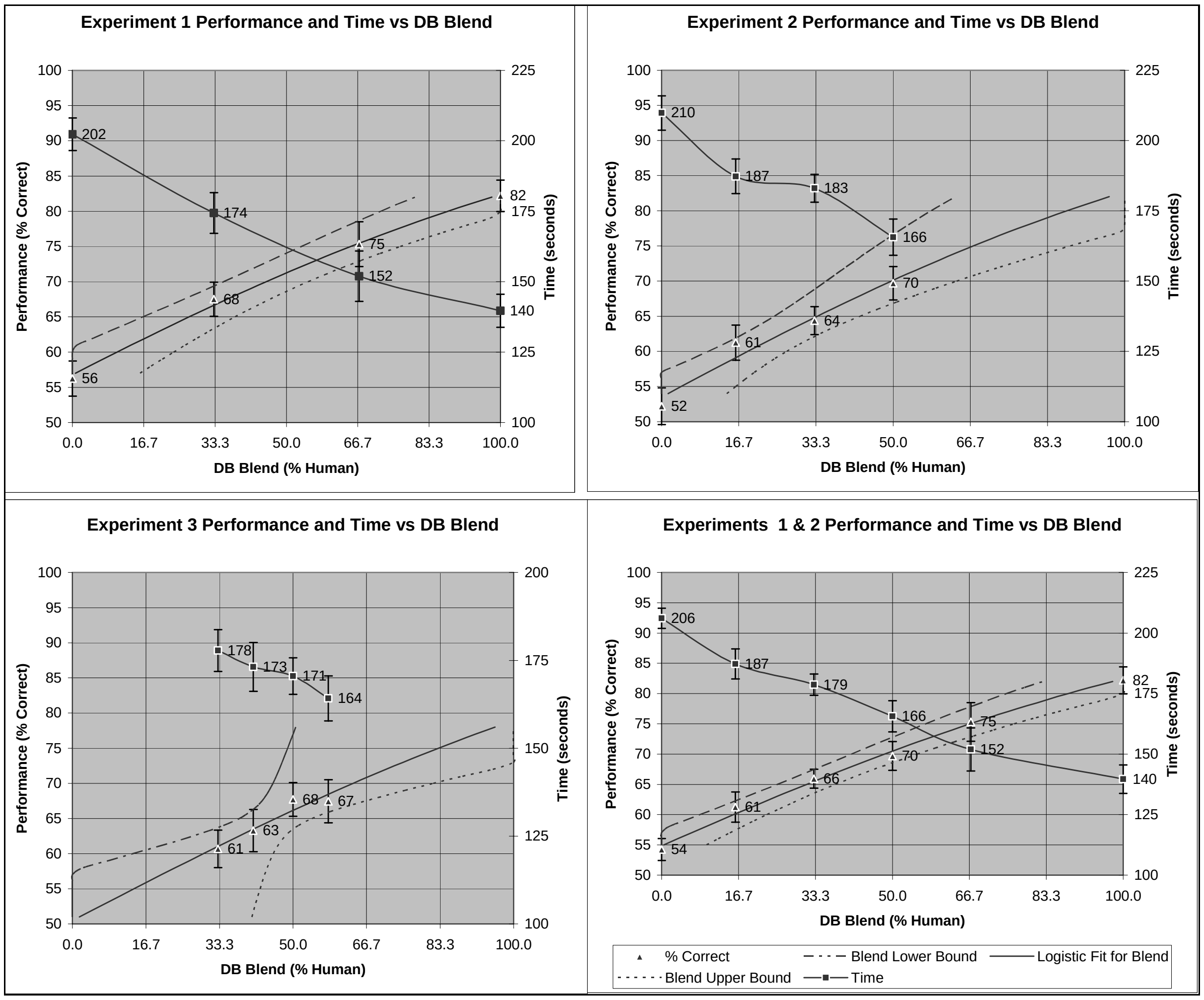


Figure 4 QA Performance (upward-sloping) and QA Time (downward-sloping) vs. Extraction Blend
Error Bars are Plus/Minus Standard Error of Mean (SEM) Within Each Blend
Upper and Lower Bounds Are Approximate 95% Confidence Intervals Based on the Logistic Fit
For the Blend (X) to Produce a Given Performance (Y)
(Read these bounds horizontally, as bounds on X, with the upper bound to the right of the lower bound.)

The downward-sloping line in each graph displays the average time to answer a question as a function of the extraction blend. For this analysis we used *strict time,* the time it took the participant to answer the question if he or she answered correctly, or the full 5 minutes allowed for any incorrectly answered question. This addresses the situation where a person quickly answers all of the questions incorrectly. The average question-answer time drops 32% as one moves from a machine generated extraction database to a human generated database. A straight-line fit to the Experiment 1 and 2 combined data predicts a drop of 6.5 seconds as the human proportion of the database increases by 10 percentage points.

A one-way repeated measures analysis of variance (ANOVA) was performed for Experiment 1 (0-33-67-100), Experiment 2 (0-16.5-33-50), and Experiment 3 (33-41-50-58). Table 5 summarizes the results. In Experiments 1 and 2 the impact of database quality on QA performance and on QA time were highly significant ($P < 0.0001$), but not for the narrower range of databases in Experiment 3. Other ANOVAs showed that the impact of trial order and question set on QA performance were both non-significant ($P > 0.05$).

Experiment	QA Performance	Strict Time
1	F(3,57) = 30.98, P < .0001	F(3, 57) = 28.36 P < .0001
2	F(3,57)= 19.32, P < .0001	F(3, 57) = 15.37, P < .0001
3	F(3,69)= 2.023, P = .1187	F(3,69)= 1.053, P = .3747

Table 5: ANOVA Analyses for QA Performance
Expt. 1 used db blends of 0, 33, 67, and 100%
Expt. 2 used db blends of 0, 16.5, 33, and 50%
Expt. 3 used db blends of 33, 41, 50, and 58%

In Experiment 1, Newman-Keuls contrasts indicate that the 0, 33, 67, and 100 databases differ significantly (P < .05) on their impact on QA quality. For Experiment 2, however, the 16.5 and 33 database qualities were not shown to be different, nor were any of the database blends in Experiment 3. The data suggest that nearly half the improvement in QA quality from 0 to 100 occurs by the 33 database blend, and more than half the improvement in QA quality from 0 to 50 occurs by the 16.5 blend: a little "human" goes a long way. Experiment 3 suggests that small differences in data blends make no practical difference in the results. Alternatively, there might be real differences that are small enough such that a larger number of participants would be required to detect them. Experiment 3 also had two participants with atypical patterns of QA against blend, which might account for the failure to detect a difference between the 33 and 50 or 58 blends as suggested by the results from Experiment 2. Furthermore, larger experiments could reveal whether the atypical participants were representatives of a subpopulation, or simply outliers. Bearing the possibility of outliers in mind, we used the combination of Experiments 1 and 2 for the combined logistic analysis.

4 Conclusions

We presented a methodology for assessing information extraction effectiveness using an extrinsic study. In addition, we demonstrated how a novel database blending (merging) strategy allows interpolating extraction quality from automated performance up through human accuracy, thereby decreasing the resources required to conduct effectiveness evaluations.

Experiments showed QA accuracy and speed increased with higher IE performance, and that the database blend percentage was a good proxy for ACE value scores. We emphasize that the study was not to show that IE supports QA better than other technologies, rather to isolate utility gains due to IE performance improvements.

QA performance was plotted against human-machine IE blend and, for example, 70% QA performance was achieved with a database blend between 41% and 46% machine extraction. This corresponded to entity and relationship value scores of roughly 74 and 47 respectively.

The logistic dose-response model provided a good fit and allowed for computation of confidence bounds for the IE associated with a particular level of performance. The constraints imposed by AnswerPad and FactBrowser ensured that world knowledge was neutralized, and the repeated-measures design (using participants as their own controls across multiple levels of database quality) excluded inter-participant variability from experimental error, increasing the ability to detect differences with relatively small sample sizes.

Acknowledgement

This material is based upon work supported in part by the Department of the Interior under Contract No. NBCHC030014. 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 Department of the Interior.

References

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel, "A Novel Use of Statistical Parsing to Extract Information from Text", in Proceedings of 1st Meeting of the North American Chapter of the ACL, Seattle, WA., pp.226-233, 2000.

S. Miller, S. Bratus, L. Ramshaw, R. Weischedel, and A. Zamanian. "FactBrowser Demonstration", Human Language Technology Conference, San Diego, 2001.

D. Jones and E. Walton, "Measuring the Utility of Human Language Technology for Intelligence Analysis," 2005 International Conference on Intelligence Applications, McLean, VA May, 2005.

Multi-Lingual Coreference Resolution With Syntactic Features

Xiaoqiang Luo and Imed Zitouni
1101 Kitchawan Road
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.
{xiaoluo, izitouni}@us.ibm.com

Abstract

In this paper, we study the impact of a group of features extracted automatically from machine-generated parse trees on coreference resolution. One focus is on designing syntactic features using the binding theory as the guideline to improve pronoun resolution, although linguistic phenomenon such as apposition is also modeled. These features are applied to the Arabic, Chinese and English coreference resolution systems and their effectiveness is evaluated on data from the Automatic Content Extraction (ACE) task. The syntactic features improve the Arabic and English systems significantly, but play a limited role in the Chinese one. Detailed analyses are done to understand the syntactic features' impact on the three coreference systems.

1 Introduction

A coreference resolution system aims to group together *mention*s referring to the same *entity*, where a mention is an instance of reference to an object, and the collection of mentions referring to the same object in a document form an *entity*. In the following example:

(I) "John believes himself to be the best student."

mentions are underlined. The three mentions "John", "himself", "the best student" are of type name, pronoun [1], and nominal, respectively. They form an *entity* since they all refer to the same person.

Syntactic information plays an important role in coreference resolution. For example, the binding theory (Haegeman, 1994; Beatrice and Kroch, 2000) provides a good account of the constraints on the antecedent of English pronouns. The theory relies on syntactic parse trees to determine the governing category which defines the scope of binding constraints. We will use the theory as a guideline to help us design features in a machine learning framework.

Previous pronoun resolution work (Hobbs, 1976; Lappin and Leass, 1994; Ge et al., 1998; Stuckardt, 2001) explicitly utilized syntactic information before. But there are unique challenges in this study: (1) Syntactic information is extracted from parse trees automatically generated. This is possible because of the availability of statistical parsers, which can be trained on human-annotated treebanks (Marcus et al., 1993; Xia et al., 2000; Maamouri and Bies, 2004) for multiple languages; (2) The binding theory is used as a guideline and syntactic structures are encoded as features in a maximum entropy coreference system; (3) The syntactic features are evaluated on three languages: Arabic, Chinese and English (one goal is to see if features motivated by the English language can help coreference resolution in other languages). All contrastive experiments are done on publicly-available data; (4) Our coreference system resolves coreferential relationships among all the annotated mentions, not just for pronouns.

Using machine-generated parse trees eliminates the need of hand-labeled trees in a coreference system. However, it is a major challenge to extract useful information from these noisy parse trees. Our approach is encoding the structures contained in a parse tree into a set of computable features, each of which is associated with a weight automatically determined by a machine learning algorithm. This contrasts with the approach of extracting rules and assigning weights to these rules by hand (Lappin and Leass, 1994; Stuckardt, 2001). The advantage of our approach is robustness: if a particular structure is helpful, it will be assigned a high weight; if a feature is extracted from a highly noisy parse tree and is not informative in coreference resolution, it will be assigned a small weight. By avoiding writing rules, we automatically incorporate useful information into our model and at the same time limit the potentially negative impact from noisy parsing output.

[1]"Pronoun" in this paper refers to both anaphor and normal pronoun.

2 Statistical Coreference Resolution Model

Our coreference system uses a binary entity-mention model $P_L(\cdot|e,m)$ (henceforth "link model") to score the action of linking a mention m to an entity e. In our implementation, the link model is computed as

$$P_L(L=1|e,m) \approx \max_{m' \in e} \hat{P}_L(L=1|e,m',m), \quad (1)$$

where m' is one mention in entity e, and the basic model building block $\hat{P}_L(L=1|e,m',m)$ is an exponential or maximum entropy model (Berger et al., 1996):

$$\hat{P}_L(L|e,m',m) = \frac{\exp\left\{\sum_i \lambda_i g_i(e,m',m,L)\right\}}{Z(e,m',m)}, \quad (2)$$

where $Z(e,m',m)$ is a normalizing factor to ensure that $\hat{P}_L(\cdot|e,m',m)$ is a probability, $\{g_i(e,m',m,L)\}$ are features and $\{\lambda_i\}$ are feature weights.

Another *start* model is used to score the action of creating a new entity with the current mention m. Since starting a new entity depends on all the partial entities created in the history $\{e_i\}_{i=1}^t$, we use the following approximation:

$$P_S(S=1|e_1,e_2,\cdots,e_t,m) \approx 1 - \max_{1\le i\le t} P_L(L=1|e_i,m) \quad (3)$$

In the maximum-entropy model (2), feature (typically binary) functions $\{g_i(e,m',m,\cdot)\}$ provide us with a flexible framework to encode useful information into the the system: it can be as simple as "$g_i(e,m',m,L=1)=1$ if m' and m have the same surface string," or "$g_j(e,m',m,L=0)=1$ if e and m differ in number," or as complex as "$g_l(e,m',m,L=1)=1$ if m' c-commands m and m' is a NAME mention and m is a pronoun mention." These feature functions bear similarity to rules used in other coreference systems (Lappin and Leass, 1994; Mitkov, 1998; Stuckardt, 2001), except that the feature weights $\{\lambda_i\}$ are automatically trained over a corpus with coreference information. Learning feature weights automatically eliminates the need of manually assigning the weights or precedence of rules, and opens the door for us to explore rich features extracted from parse trees, which is discussed in the next section.

3 Syntactic Features

In this section, we present a set of features extracted from syntactic parse trees. We discuss how we *approximately* compute linguistic concepts such as governing category (Haegeman, 1994), apposition and dependency relationships from noisy syntactic parse trees. While parsing and parse trees depend on the target language, the automatic nature of feature extraction from parse trees makes the process language-independent.

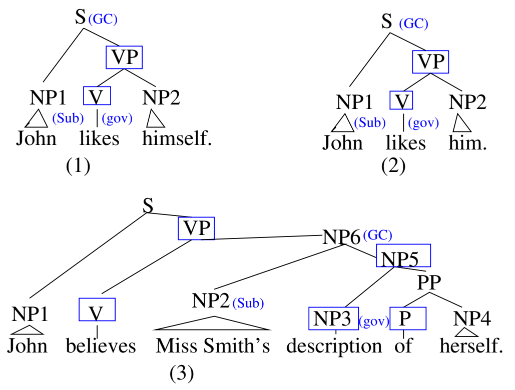


Figure 1: GC examples.

3.1 Features Inspired by Binding Theory

The binding theory (Haegeman, 1994) concerning pronouns can be summarized with the following principles:

1. A reflexive or reciprocal pronoun (e.g., "herself" or "each other") must be bound in its governing category (GC).
2. A normal pronoun must be free in its governing category.

The first principle states that the antecedent of a reflexive or reciprocal pronoun is within its GC, while the second principle says that the antecedent of a normal pronoun is outside its GC. While the two principles are simple, they all rely on the concept of governing category, which is defined as the minimal domain containing the pronoun in question, its governor, and an accessible subject.

The concept GC can best be explained with a few examples in Figure 1, where the label of a head constituent is marked within a box, and GC, accessible subject, and governor constituents are marked in parentheses with "GC", "Sub" and "gov." Noun-phrases (NP) are numbered for the convenience of referencing. For example, in sub-figure (1) of Figure 1, the governor of "himself" is "likes," the subject is "John," hence the GC is the entire sentence spanned by the root "S." Since "himself" is reflexive, its antecedent must be "John" by Principle 1. The parse tree in sub-figure (2) is the same as that in sub-figure (1), but since "him" is a normal pronoun, its antecedent, according to Principle 2, has to be outside the GC, that is, "him" cannot be coreferenced with "John.". Sentence in sub-figure (3) is slightly more complicated: the governor of "herself" is "description," and the accessible subject is "Miss Smith." Thus, the governing category is NP6. The first principle implies that the antecedent of "herself" must be "Miss Smith."

It is clear from these examples that GC is very useful in finding the antecedent of a pronoun. But the last example shows that determining GC is not a trivial matter. Not only is the correct parse tree required, but extra information is also needed to identify the head governor and the minimal constituent dominating the pronoun, its governor and an accessible subject. Determining the accessible subject itself entails checking other constraints such as number and gender agreement. The complexity of computing governing category, compounded with the noisy nature of machine-generated parse tree, prompts us to compute a set of features that characterize the structural relationship between a candidate mention and a pronoun, as opposed to explicitly identify GC in a parse tree. These features are designed to implicitly model the binding constraints.

Given a candidate antecedent or mention m_1 and a pronoun mention m_2 within a parsed sentence, we first test if they have c-command relation, and then a set of counting features are computed. The features are detailed as follows:

(1) C-command $ccmd(m_1, m_2)$: A constituent X c-commands another constituent Y in a parse tree if the first branching node dominating X also dominates Y. The binary feature $ccmd(m_1, m_2)$ is true if the minimum NP dominating m_1 c-commands the minimum NP dominating m_2. In sub-figure (1) of Figure 1, NP1 c-commands NP2 since the first branching node dominating NP1 is S and it dominates NP2.

If $ccmd(m_1, m_2)$ is true, we then define the c-command path $T(m_1, m_2)$ as the path from the minimum NP dominating m_2 to the first branching node that dominates the minimum NP dominating m_1. In sub-figure (1) of Figure 1, the c-command path T("John", "himself") would be "NP2-VP-S."

(2) $NP\_count(m_1, m_2)$: If $ccmd(m_1, m_2)$ is true, then $NP\_count(m_1, m_2)$ counts how many NPs are seen on the c-command path $T(m_1, m_2)$, excluding two endpoints. In sub-figure (1) of Figure 1, NP_count("John", "himself") $= 0$ since there is no NP on T("John", "himself").

(3) $VP\_count(m_1, m_2)$: similar to $NP\_count(m_1, m_2)$, except that this feature counts how many verb phrases (VP) are seen on the c-command path. In sub-figure (1) of Figure 1, VP_count("John", "himself") is true since there is one VP on T("John", "himself").

(4) $S\_count(m_1, m_2)$: This feature counts how many clauses are seen on the c-command path when $ccmd(m_1, m_2)$ is true. In sub-figure (1) of Figure 1, S_count("John", "himself") $= 0$ since there is no clause label on T("John", "himself").

These features are designed to capture information in the concept of governing category when used in conjunction with attributes (e.g., gender, number, reflexiveness) of individual pronouns. Counting the intermediate NPs, VPs and sub-clauses implicitly characterizes the governor of a pronoun in question; the presence or absence of a sub-clause indicates whethere or not a coreferential relation is across clause boundary.

3.2 Dependency Features

In addition to features inspired by the binding theory, a set of dependency features are also computed with the help of syntactic parse trees. This is motivated by examples such as "John is the president of ABC Corporation," where "John" and "the president" refer to the same person and should be in the same entity. In scenarios like this, lexical features do not help, while the knowledge that "John" left-modifies the verb "is" and the "the president" right-modifies the same verb would be useful.

Given two mentions m_1 and m_2 in a sentence, we compute the following dependency features:

(1)$same\_head(m_1, m_2)$: The feature compares the bilexical dependencies $\langle m_1, h(m_1)\rangle$, and $\langle m_2, h(m_2)\rangle$, where $h(x)$ is the head word which x modifies. The feature is active only if $h(m_1) = h(m_2)$, in which case it returns $h(m_1)$.

(2)$same\_POS(m_1, m_2)$: To get good coverage of dependencies, we compute a feature $same\_POS(m_1, m_2)$, which examines the same dependency as in (1) and returns the common head part-of-speech (POS) tag if $h(m_1) = h(m_2)$.

The head child nodes are marked with boxes in Figure 1. For the parse tree in sub-figure (1), $same\_head$("John", "him") would return "likes" as "John" left-modifies "likes" while "him" right-modifies "likes," and $same\_POS$("John", "him") would return "V" as the POS tag of "likes" is "V."

(3) $mod(m_1, m_2)$: the binary feature is true if m_1 modifies m_2. For parse tree (2) of Figure 1, mod("John", "him") returns false as "John" does not modify "him" directly. A reverse order feature $mod(m_2, m_1)$ is computed too.

(4) $same\_head2(m_1, m_2)$: this set of features examine second-level dependency. It compares the head word of $h(m_1)$, or $h(h(m_1))$, with $h(m_2)$ and returns the common head if $h(h(m_1)) = h(m_2)$. A reverse order feature $same\_head2(m_2, m_1)$ is also computed.

(5) $same\_POS2(m_1, m_2)$: similar to (4), except that it computes the second-level POS. A reverse order feature $same\_POS2(m_2, m_1)$ is computed too.

(6) $same\_head22(m_1, m_2)$: it returns the common second-level head if $h(h(m_1)) = h(h(m_2))$.

3.3 Apposition and Same-Parent Features

Apposition is a phenomenon where two adjacent NPs refer to the same entity, as "Jimmy Carter" and "the former president" in the following example:

(II) "$\underline{\text{Jimmy Carter}}$, $\underline{\text{the former president}}$ of US, is visiting Europe."

Note that not all NPs separated by a comma are necessarily appositive. For example, in "John called Al, Bob, and Charlie last night," "Al" and "Bob" share a same NP parent and are separated by comma, but they are not appositive.

To compute the apposition feature $appos(m_1, m_2)$ for mention-pair (m_1, m_2), we first determine the minimum dominating NP of m_1 and m_2. The minimum dominating NP of a mention is the lowest NP, with an optional modifying phrase or clause, that spans the mention. If the two minimum dominating NPs have the same parent NP, and they are the only two NP children of the parent, the value of $appos(m_1, m_2)$ is true. This would exclude "Al" and "Bob" in "John called Al, Bob, and Charlie last night" from being computed as apposition.

We also implement a feature $same\_parent(m_1, m_2)$ which tests if two mentions m_1 and m_2 are dominated by a common NP. The feature helps to prevent the system from linking "his" with "colleague" in the sentence "John called his colleague."

All the features described in Section 3.1-3.3 are computed from syntactic trees generated by a parser. While the parser is language dependent, feature computation boils down to encoding the structural relationship of two mentions, which is language independent. To test the effectiveness of the syntactic features, we integrate them into 3 coreference systems processing Arabic, Chinese and English.

4 Experimental Results

4.1 Data and System Description

All experiments are done on true mentions of the ACE (NIST, 2004) 2004 data. We reserve part of LDC-released 2004 data as the development-test set (henceforth "devtest") as follows: documents are sorted by their date and time within each data source (e.g., broadcast news (bnews) and news wire (nwire) are two different sources) and the last 25% documents of each data source are reserved as the devtest set. Splitting data on chronological order simulates the process of a system's development and deployment in the real world. The devtest set statistics of three languages (Arabic, Chinese and English) is summarized in Table 1, where the number of documents, mentions and entities is shown on row 2 through 4, respectively. The rest of 2004 ACE data together with earlier ACE data is used as training.

	Arabic	Chinese	English
#-docs	178	166	114
#-mentions	11358	8524	7008
#-entities	4428	3876	2929

Table 1: Devtest Set Statistics by Language

The official 2004 evaluation test set is used as the blind test set on which we run our system once after the system development is finished. We will report summary results on this test set.

As for parser, we train three off-shelf maximum-entropy parsers (Ratnaparkhi, 1999) using the Arabic, Chinese and English Penn treebank (Maamouri and Bies, 2004; Xia et al., 2000; Marcus et al., 1993). Arabic words are segmented while the Chinese parser is a character-based parser. The three parsers have a label F-measure of 77%, 80%, and 86% on their respective test sets. The three parsers are used to parse both ACE training and test data. Features described in Section 3 are computed from machine-generated parse trees.

Apart from features extracted from parse trees, our coreference system also utilizes other features such as lexical features (e.g., string matching), distance features characterized as quantized word and sentence distances, mention- and entity-level attribute information (e.g, ACE distinguishes 4 types of mentions: NAM(e), NOM(inal), PRE(modifier) and PRO(noun)) found in the 2004 ACE data. Details of these features can be found in (Luo et al., 2004).

4.2 Performance Metrics

The official performance metric in the ACE task is ACE-Value (NIST, 2004). The ACE-Value is an entity-based metric computed by subtracting a normalized cost from 1 (so it is unbounded below). The cost of a system is a weighted sum of costs associated with entity misses, false alarms and errors. This cost is normalized against the cost of a nominal system that outputs no entity. A perfect coreference system gets 100% ACE-Value while a system outputting many false-alarm entities could get a negative value.

The default weights in ACE-Value emphasize names, and severely discount pronouns: the relative importance of a pronoun is two orders of magnitude less than that of a name. So the ACE-Value will not be able to accurately reflect a system's improvement on pronouns[2]. For this reason, we compute an *unweighted* entity-constrained mention F-measure (Luo, 2005) and report all contrastive experiments with this metric. The F-measure is computed by first aligning system and reference entities such that the number of common mentions is maximized and each system entity is constrained to align with at most one reference entity, and vice versa. For example, suppose that a reference document contains three entities: $\{[m_1], [m_2, m_3], [m_4]\}$ while a system outputs four entities: $\{[m_1, m_2], [m_3], [m_5], [m_6]\}$, where $\{m_i : i = 1, 2, \cdots, 6\}$ are mentions, then the best alignment from reference to system would be $[m_1] \Leftrightarrow [m_1, m_2]$, $[m_2, m_3] \Leftrightarrow [m_3]$ and other entities are not aligned. The number of common mentions of the best alignment is 2

[2]Another possible choice is the MUC F-measure (Vilain et al., 1995). But the metric has a systematic bias for systems generating fewer entities (Bagga and Baldwin, 1998) – see Luo (2005). Another reason is that it cannot score single-mention entity.

(i.e., m_1 and m_3), thus the recall is $\frac{2}{4}$ and precision is $\frac{2}{5}$. Due to the one-to-one entity alignment constraint, the F-measure here is more stringent than the accuracy (Ge et al., 1998; Mitkov, 1998; Kehler et al., 2004) computed on antecedent-pronoun pairs.

4.3 Effect of Syntactic Features

We first present the contrastive experimental results on the devtest described in sub-section 4.1.

Two coreference systems are trained for each language: a baseline without syntactic features, and a system including the syntactic features. The entity-constrained F-measures with mention-type breakdown are presented in Table 2. Rows marked with N_m contain the number of mentions, while rows with "base" and "+synt" are F-measures for the baseline and the system with the syntactic features, respectively.

The syntactic features improve pronoun mentions across three languages – not surprising since features inspired by the binding theory are designed to improve pronouns. The pronoun improvement on the Arabic (from 73.2% to 74.6%) and English (from 69.2% to 72.0%) system is statistically significant (at above 95% confidence level), but change on the Chinese system is not. For Arabic, the syntactic features improve Arabic NAM, NOM and PRE mentions, probably because Arabic pronouns are sometimes attached to other types of mentions. For Chinese and English, the syntactic features do not practically change the systems' performance.

As will be shown in Section 4.5, the baseline systems without syntactic features are already competitive, compared with the results on the coreference evaluation track (EDR-coref) of the ACE 2004 evaluation (NIS, 2004). So it is nice to see that syntactic features further improve a good baseline on Arabic and English.

	Mention Type				
	NAM	NOM	PRE	PRO	Total
Arabic					
N_m	2843	3438	1291	3786	11358
base	86.8	73.2	86.7	73.2	78.2
+synt	**88.4**	**76.4**	**87.4**	**74.6**	**80.1**
Chinese					
N_m	4034	3696	-	794	8524
base	**95.4**	**77.8**	-	65.9	**85.0**
+synt	95.2	77.7	-	**66.5**	84.9
English					
N_m	2069	2173	835	1931	7008
base	92.0	73.4	**88.7**	69.2	79.6
+synt	92.0	**75.3**	87.8	**72.0**	**80.8**

Table 2: F-measure(%) Breakdown by Mention Type: NAM(e), NOM(inal), PRE(modifier) and PRO(noun). Chinese data does not have the PRE type.

4.4 Error Analyses

From the results in Table 2, we know that the set of syntactic features are working in the Arabic and English system. But the results also raise some questions: Are there interactions among the the syntactic features and other features? Why do the syntactic features work well for Arabic and English, but not Chinese? To answer these questions, we look into each system and report our findings in the following sections.

4.4.1 English System

Our system uses a group of distance features. One observation is that information provided by some syntactic features (e.g., $VP\_count(m_1, m_2)$ etc) may have overlapped with some of the distance features. To test if this is the case, we take out the distance features from the English system, and then train two systems, one with the syntactic features, one without. The results are shown in Table 3, where numbers on the row "b-dist" are F-measures after removing the distance features from the baseline, and numbers on the row "b-dist+synt" are with the syntactic features.

	Mention Type				
	NAM	NOM	PRE	PRO	Total
b-dist	84.2	68.8	74.6	63.3	72.5
b-dist+synt	**90.7**	**74.2**	**87.8**	**69.0**	**79.3**

Table 3: Impact of Syntactic Features on English System After Taking out Distance Features. Numbers are F-measures(%).

As can be seen, the impact of the syntactic features is much larger when the distance features are absent in the system: performance improves across all the four mention types after adding the syntactic features, and the overall F-measure jumps from 72.5% to 79.3%. The PRE type gets the biggest improvement since features extracted from parse trees include apposition, same-parent test, and dependency features, which are designed to help mention pairs in close distance, just as in the case of PRE mentions.

Comparing the numbers in Table 3 with the English baseline of Table 2, we can also conclude that distance features and syntactic features lead to about the same level of performance when the other set of features is not used. When the distance features are used, the syntactic features further help to improve the performance of the NOM and PRO mention type, albeit to a less degree because of information overlap between the two sets of features.

4.4.2 Chinese System

Results in Table 2 show that the syntactic features are not so effective for Chinese as for Arabic and English. The first thing we look into is if there is any idiosyncrasy in the Chinese language.

In Table 4, we list the statistics collected over the training sets of the three languages: the second row are the total number of mentions, the third row the number of pronoun mentions, the fourth row the number of events where the c-command feature $ccmd(m_1, m_2)$ is used, and the last row the average number of c-command features per pronoun (i.e., the fourth row divided by the third row). A pronouns event is defined as a tuple of training instance (e, m_1, m_2) where m_1 is a mention in entity e, and the second mention m_2 is a pronoun.

From Table 4, it is clear that Chinese pronoun distribution is very different: pronoun mentions account for about 8.7% of the total mentions in Chinese, while 29.0% of Arabic mentions and 25.1% of English mentions are pronouns (the same disparity can be observed in the devtest set in Table 2). This is because Chinese is a pro-drop language (Huang, 1984): for example, in the Chinese Penn treebank version 4, there are 4933 overt pronouns, but 5750 pro-drops! The ubiquity of pro-drops in Chinese results in signigicantly less pronoun training events. Consequently, the pronoun-related features are not trained as well as in English and Arabic. One way to quantify this is by looking at the average number of c-command features on a per-pronoun basis: as shown in the last row of Table 4, the c-command feature is seen more than twice often in Arabic and English as in Chinese. Since low-count features are filtered out, the sparsity of pronoun events prevent many compound features (e.g., conjunction of syntactic and distance features) from being trained in the Chinese system, which explains why the syntactic features do not help Chinese pronouns.

	Arabic	Chinese	English
#total-mentions	31706	33851	58202
#pron-mentions	9183	2941	14635
#-ccmd-event	10236	1260	13691
#ccmd/pron	1.14	0.428	0.936

Table 4: Distribution of Pronoun Mentions and Frequency of c-command Features

4.4.3 Arabic System

As stated in Table 4, 29.0% of Arabic mentions are pronouns, compared to a slightly lower number (25.1%) for English. This explains the relatively high positive impact of the syntactic features on the Arabic coreference system, compared to English and Chinese systems. To understand how syntactic features work in the Arabic system, we examine two examples extracted from the devtest set: (1) the first example shows the negative impact of syntactic features because of the noisy parsing output, and (2) the second example proves the effectiveness of the syntactic features to find the dependency between two mentions. In both examples, the baseline system and the system with syntactic features give different results.

Let's consider the following sentence:

... وتعتبر إسرِئيل **القدس** عَاصمتها ...

... its-capital ← Jerusalem ← Israel ← consider ← and ...

... فيمَا يريد الفلسطينيون الشطر الشرقي **لِلمدينة** ...

of-the-city ← the-Eastern ← the-half ← the-Palestininan ← want ← while

The English text shown above is a word-to-word translation of the Arabic text (read from right-to-left). In this example, the parser wrongly put the nominal mention القدس (Jerusalem) and the pronominal mention المدينة (the-city) under the same constituent, which activates the *same_parent* feature. The use of the feature *same_parent*(القدس, المدينة) leads to the two mentions being put into different entities. This is because there are many cases in the training data where two mentions under the same parent are indeed in different entities: a similar English example is "*John called his sister*", where "*his*" and "*sister*" belong to two different entities. The *same_parent* feature is a strong indicator of not putting them into the same entity.

كَان + **ال** + **زقَاقيون** + ي + حَاول + ون + نهب + ال + محل + ات + ال + تجَاري + ة ب + **حجة** + أَن + **هم** + ...
kAn + **Al** + **zqAqywn** + y + HAwl + wn + nhb + Al + mHl + At + Al + tjAry + p + b + Hjp + An + **hm** + ...
was + **the** + **zqAqywn** + *present-verb-marker* y + trying + *plural-verb-marker* wn + to-steal + the + office + s + the + commercial + s + with + excuse + that + **they** + ...

Table 5: An example where syntactic features help to link the PRO mention هم (hm) with its antecedent, the NAM mention الزقاقيون (AlzqAqywn): top – Arabic sentence; middle – corresponding romanized sentence; bottom – token-to-token English translation.

Table 5 shows another example in the devtest set. The top part presents the segmented Arabic text, the middle part is the corresponding romanized text, and the bottom part contains the token-to-token English translation. Note that Arabic text reads from right to left and its corresponding romanized text from left to right (i.e., the right-most Arabic token maps to the left-most romanized token). The parser output the correct syntactic structure: Figure 2 shows a portion of the system-generated parse tree. It can be checked that NP1 c-commands NP2 and the group of features inspired by the binding theory are active. These features help to link the PRO(onominal) mention هم (hm) with the NAM(e) mention الزقاقيون (AlzqAqywn). Without syntactic features theses two mentions were split into different entities.

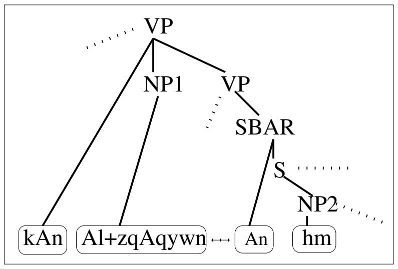


Figure 2: A Portion of the Syntactic Tree.

4.5 ACE 2004 Results

To get a sense of the performance level of our system, we report the results on the ACE 2004 official test set with both the F-measure and the official ACE-Value metric. This data is used as the blind test set which we run our system only once.

Results are summarized in Table 6, where the second row (i.e. "base") contains the baseline numbers, and the third row (i.e., "+synt") contains the numbers from systems with the syntactic features. Columns under "F" are F-measure and those under "AV" are ACE-Value. The last row N_m contains the number of mentions in the three test sets.

	Arabic		Chinese		English	
	F	AV	F	AV	F	AV
base	80.1	88.0	84.7	92.7	80.6	90.9
+synt	81.5	88.9	84.7	92.8	82.0	91.6
N_m	11358		11178		10336	

Table 6: Summary Results on the 2004 ACE Evaluation Data.

The performance of three full ("+synt") systems is remarkably close to that on the devtest set(cf. Table 2): For Arabic, F-measure is 80.1 on the devtest vs. 81.5 here; For Chinese, 84.9 vs. 84.7; And for English, 80.8 vs. 82.0. The syntactic features again help Arabic and English – statistically very significant in F-measure, but have no significant impact on Chinese. The performance consistency across the devtest and blind test set indicates that the systems are well trained.

The F-measures are computed on all types of mentions. Improvement on mention-types targeted by the syntactic features is larger than the lump-sum F-measure. For example, the F-measure for English pronouns on this test set is improved from 69.5% to 73.7% (not shown in Table 6 due to space limit). The main purpose of Table 6 is to get a sense of performance level correspondence between the F-measure and ACE-Value.

Also note that, for Arabic and English, the difference between the "base" and "+synt" systems, when measured by ACE-Value, is much smaller. This is not surprising since ACE-Value heavily discounts pronouns and is insensitive to improvement on pronouns – the very reason we adopt the F-measure in Section 4.3 and 4.4 when reporting the contrastive experiment results.

5 Related Work

Many researchers have used the syntactic information in their coreference system before. For example, Hobbs (1976) uses a set of rules that are applied to parse trees to determine the antecedent of a pronoun. The rule precedence is determined heuristically and no weight is used. Lappin and Leass (1994) extracted rules from the output of the English Slot Grammar (ESG) (McCord, 1993). Rule weights are assigned manually and the system resolves the third person pronouns and reflexive pronouns only. Ge et al. (1998) uses a non-parametrized statistical model to find the antecedent from a list of candidates generated by applying the Hobbs algorithm to the English Penn Treebank. Kehler et al. (2004) experiments making use of predicate-argument structure extracted from a large TDT-corpus. Compared with these work, our work uses machine-generated parse trees from which trainable features are extracted in a maximum-entropy coreference system, while (Ge et al., 1998) assumes that correct parse trees are given. Feature weights are automatically trained in our system while (Lappin and Leass, 1994; Stuckardt, 2001) assign weights manually.

There are a large amount of published work (Morton, 2000; Soon et al., 2001; Ng and Cardie, 2002; Yang et al., 2003; Luo et al., 2004; Kehler et al., 2004) using machine-learning techniques in coreference resolution. But none of these work tried to compute complex linguistic concept such as governing category [3]. Our work demonstrates how relevant linguistic knowledge can be derived automatically from system-generated parse trees and encoded into computable and trainable features in a machine-learning framework.

6 Conclusions

In this paper, linguistic knowledge is used to guide us to design features in maximum-entropy-based coreference resolution systems. In particular, we show how to compute a set of features to approximate the linguistic notions such as governing category and apposition, and how to compute the dependency features using syntactic parse trees. While the features are motivated by examining English data, we see significant improvements on both English and Arabic systems. Due to the language idiosyncrasy (e.g., pro-drops), we do not see the syntactic features change the Chinese system significantly.

[3]Ng and Cardie (2002) used a BINDING feature, but it is not clear from their paper how the feature was computed and what its impact was on their system.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred.

Suggestions for improving the paper from the anonymous reviewers are gratefully acknowledged.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at The First International Conference on Language Resources and Evaluation (LREC'98)*, pages 563–566.

Santorini Beatrice and Anthony Kroch. 2000. *The syntax of natural language: An online introduction using the Trees program*. www.ling.upenn.edu/ beatrice/syntax-textbook.

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, March.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proc. of the sixth Workshop on Very Large Corpora*.

Liliane Haegeman. 1994. *Introduction to Government and Binding Theory*. Basil Blackwell Inc., 2nd edition.

J. Hobbs. 1976. Pronoun resolution. Technical report, Dept. of Computer Science, CUNY, Technical Report TR76-1.

C.-T. James Huang. 1984. On the distribution and reference of empty pronouns. *Linguistic Inquiry*, 15:531–574.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. The (Non)utility of predicate-argument frequencies for pronoun interpretation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

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

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proc. of ACL*.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Procs. of HLT/EMNLP*.

Mohamed Maamouri and Ann Bies. 2004. Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, COLING*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

Michael McCord. 1993. Heuristics for broad-coverage natural language parsing. In *Proc. ARPA Human Language Technology Workshop*.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Procs. of the 36th ACL/17th COLING*, pages 869–875.

Thomas S. Morton. 2000. Coreference for NLP applications. In *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*, pages 104–111.

NIST. 2004. *Proceedings of ACE Evaluation and PI Meeting 2004 Workshop*, Alexandria, VA, September.

NIST. 2004. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–178.

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

Roland Stuckardt. 2001. Design and enhanced evaluation of a robust anaphor resolution algorithm. *Computational Linguistics*, 27(4).

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, , and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *In Proc. of MUC6*, pages 45–52.

F. Xia, M. Palmer, N. Xue, M.E. Okurowski, J. Kovarik, F.D. Chiou, S. Huang, T. Kroch, and M. Marcus. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proc of the 2nd Intl. Conf. on Language Resources and Evaluation (LREC 2000)*.

Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proc. of ACL*.

Analyzing models for semantic role assignment using confusability

Katrin Erk and **Sebastian Padó**
Computational Linguistics
Saarland University
Saarbrücken, Germany
{erk,pado}@coli.uni-sb.de

Abstract

We analyze models for semantic role assignment by defining a *meta-model* that abstracts over features and learning paradigms. This meta-model is based on the concept of *role confusability*, is defined in information-theoretic terms, and predicts that roles realized by less specific grammatical functions are more difficult to assign. We find that confusability is strongly correlated with the performance of classifiers based on syntactic features, but not for classifiers including semantic features. This indicates that syntactic features approximate a description of grammatical functions, and that semantic features provide an independent second view on the data.

1 Introduction

Semantic roles have become a focus of research in computational linguistics during the recent years. The driving force behind this interest is the prospect that semantic roles, as a shallow meaning representation, can improve many NLP applications, while still being amenable to automatic analysis. The benefit of semantic roles has already been demonstrated for a number of tasks, among others for machine translation (Boas, 2002), information extraction (Surdeanu et al., 2003), and question answering (Narayanan and Harabagiu, 2004).

Robust and accurate *automatic semantic role assignment*, a prerequisite for the wide-range use of semantic roles in NLP, has been investigated in a number of studies and shared tasks. Typically, role assignment has been modeled as a classification task, with models being estimated from large corpora (Gildea and Jurafsky, 2002; Moschitti, 2004; Xue and Palmer, 2004; Surdeanu et al., 2003; Pradhan et al., 2004; Litkowski, 2004; Carreras and Màrquez, 2005).

Within this framework, there is a number of architectural parameters which lend themselves to optimization: the machine learning framework, the feature set, pre- and postprocessing, each of which has been investigated in the context of semantic role assignment. The current paper concentrates on feature engineering, since the feature set is a pivotal component of any kind of machine learning system, and allows us to incorporate and test linguistic intuitions on the role assignment task.

We approach feature engineering not by directly optimizing system performance. Instead, we proceed by *error analysis*, like Pado and Boleda (2004). Our aim is to form a *global hypothesis* that explains the distribution of errors across classes. Insofar as the model does not contain model-specific information, following this methodology can provide a *meta-model* of a model family which abstracts over concrete features and over the learning paradigm.

The concrete global hypothesis we test is: (1) All features of current models approximate a description of grammatical functions, and the complete systems approximate an assignment based on grammatical functions. (2) System performance for a given role depends on how easily it is confused with other roles. We will give this concept of *role confusability* a formal, information-theoretic definition.

The present study specifically analyzes models for semantic role assignment in the FrameNet paradigm (Fillmore et al., 2003). We are going to show that our hypothesis indeed holds for a variety of models – but only models that comprise exclusively syntactic features. We conclude that syntactic features approximate a description of grammatical functions, but that semantic features model a different aspect of the role assignment mapping. Together with the reasonable performance of a solely semantics-based system, this leads us to suggest a closer investigation of semantic features – and in particular, a co-training approach with syntactic and semantic features as different views on the role assignment data.

Plan of the paper. In Section 2, we give a brief introduction to FrameNet, the semantic role paradigm and corpus we are using in this study. Our first experiment, described in Section 3, establishes that there is a high variance in performance across roles, and that this variance is itself stable across models and learners. In Section 4, we state our hypothesis, namely that this variance can be explained through role confusability, and formalize the concept . In Section 5, we perform detailed correlation tests to verify our hypothesis and discuss our findings. Section 6 concludes the paper.

2 FrameNet

This section presents the semantic role paradigm and the role-annotated corpus on which the present study is based. FrameNet[1] is a lexical resource based on Fillmore's Frame Semantics (Fillmore, 1985). It describes *frames*, representations of prototypical situations. Each frame provides its set of semantic roles, the entities or concepts pertaining to the prototypical situation. Each frame is further associated with a set of *target predicates* (nouns, verbs or adjectives), occurrences of which can introduce the frame.

FrameNet provides manually annotated examples for each predicate, sampled from the British National Corpus (Burnard, 1995). The size of this corpus exceeds 135,000 sentences. The following sentences are examples for verbs in the IMPACT frame, which describes a situation in which typically "an IMPACTOR makes sudden, forcible contact with the IMPACTEE, or two IMPACTORS both ... [make] forcible contact":

[1]`http://www.icsi.berkeley.edu/~framenet/`

(1) [$_{\text{Impactee}}$ His car] was **struck** [$_{\text{Impactor}}$ by a third vehicle].

(2) [$_{\text{Impactor}}$ The door] **slammed** [$_{\text{Result}}$ shut].

(3) [$_{\text{Impactors}}$ Their vehicles] **collided** [$_{\text{Place}}$ at Pond Hill].

FrameNet manual annotation also comprises a layer of grammatical functions: For example, the subject of finite verbs is labeled `Ext`, and `Mod` is a label used for modifiers of heads, e.g. an adjective modifying a noun. The grammatical functions used in FrameNet are listed in Fillmore and Petruck (2003).

Note that the frame-specificity of semantic roles in FrameNet has important consequences for semantic role assignment, since there is no direct way to generalize role assignments across frames, and learning has to proceed frame-wise. This compounds the data sparseness problem, and automatic assignment for frames with no training data is very difficult (Gildea and Jurafsky, 2002).

3 Experiment 1: Variance in role assignment

Several studies have established that there is considerable variance in semantic role assignment performance across different semantic roles within systems (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005; Pado and Boleda Torrent, 2004). However, these studies used either the PropBank semantic role paradigm (Carreras and Màrquez) or a limited of experimental conditions (Pado and Boleda). For this reason, we perform a first experiment to replicate this phenomenon in our setting.

Note that the vast majority of participant systems in recent shared tasks divides semantic role assignment into multiple sequential steps. The maximal decomposition is as follows: *preprocessing*, e.g. removal of unlikely argument candidates; *argument recognition*, the distinction between role-bearing and non-role-bearing instances; *argument labeling*, the actual classification of role-bearing instances; and *postprocessing*, e.g. by inference over probable role sequences.

Following this distinction, we concentrate in this study on the argument labeling step, i.e. distinguishing between roles, rather than distinguishing roles from non-roles. This is justified by earlier empirical results, namely that the argument labeling step requires more training data than argument recognition (Fleischmann and Hovy, 2003), and that it calls for more sophisticated feature construction (Xue and Palmer, 2004). We take this as evidence that the quality of the argument labeling step is central to a good semantic role assignment system.

In order to isolate the effects of argument labeling, we assume perfect argument recognition by using gold standard role boundaries; however, we do not use gold standard parse trees, but rather automatically computed ones, which realistically introduces some noise (see the following paragraph).

Data and preprocessing. As experimental material, we used the same data that was used in the Senseval-3 semantic role assignment task: 40 frames from FrameNet version 1.1, comprising 66,777 instances. The number of roles per frame ranged from 2 to 22, and the number of role instances ranged from 593 to 8,378. The data was randomly split into training (90%) and test instances (10%).

The data was parsed with the Collins model 3 (1996) parser; in addition, all tokens were lemmatized with TreeTagger (Schmid, 1994).

Modeling. We model role assignment as a classification task, with parse tree constituents as instances to be classified. We repeated the classification with two different learners: The first learner, TiMBL (Daelemans et al., 2003) is an implementation of nearest-neighbor classification algorithms in the memory-based learning paradigm[2]. The second learner, Malouf's probabilistic maximum entropy (Maxent) system (Malouf, 2002), uses the LMVM algorithm to estimate log-linear models. We did not perform smoothing.

Table 5 shows the features we use. Here as in the system setup, we keep close to current existing models for semantic role assignment in order to make our results as representative as possible. We investigate different feature sets in order to verify our results. In Exp. 1, we limit ourselves to two feature sets, *Syn* (syntactic features) and *Sem* (lexical features) from the bottom of Table 5. The feature sets were exactly the same for both learners.

[2]TiMBL was set to k-NN classification, using the MVDM distance metric and 5 neighbors.

	Syn/Sem	Syn
MBL	87.1 ± 12.7	82.2 ± 17.8
Maxent	87.5 ± 13.4	$82.4 \pm 18.\ 2$

Table 1: Exp. 1: Overall results (F-scores and standard deviation across roles).

	Syn/Sem		Syn	
Role	F_{MBL}	F_{Maxent}	F_{MBL}	F_{Maxent}
Frame: CHANGE_POSITION_ON_A_SCALE				
ATTR	79.0	80.7	57.6	66.1
CO_VAR	55.6	64.0	22.2	31.6
DIFF	87.1	84.9	75.0	66.7
ITEM	68.6	70.3	48.0	61.3
VALUE_1	88.0	91.7	78.3	72.7
VALUE_2	93.3	90.9	89.3	85.2
Frame: KINSHIP				
ALTER	87.0	89.2	87.8	87.4
EGO	96.7	98.8	96.7	95.5
Frame: PART_ORIENTATIONAL				
PART	98.2	96.4	97.6	97.0
WHOLE	100	100	98.2	100
Frame: TRAVEL				
AREA	31.6	52.6	25.0	45.5
GOAL	74.4	71.4	68.3	62.2
MODE	46.2	72.7	12.5	15.4
PATH	66.7	53.3	50.0	40.0
SOURCE	66.7	72.7	66.7	66.7
TIME	77.8	66.7	15.4	40.0
TRAVELER	90.9	90.6	90.9	90.6

Table 2: Exp. 1: Role-specific figures of system performance for four example frames.

Results. Table 1 shows the systems' overall F-scores and standard deviation across roles. Table 2 illustrates the differences in performance across roles on four frames: It lists all roles with ≥ 5 occurrences for each frame. PART_ORIENTATIONAL shows very little variance, while the roles of CHANGE_POSITION_ON_A_SCALE and especially TRAVEL differ widely. For KINSHIP, the system shows good performance for both roles, but the F-scores still differ by around 9 points.

Discussion. Table 1 shows that there is considerable variance across roles, with a standard deviation in the range of 18% for the syntax-only model. We note that the deviation decreases to 13% for the combined syntax-semantics model. Table 2 confirms that this is not purely between-frames, but also within-frames variance. This confirms the phenomenon described at the beginning of this section.

fr	frame
fe	role (frame element)
$fes(fr)$	roles of a frame
$gfs(fr)$	gramm. functions of a frame
$gfs_{fr}(fe)$	gramm. functions realizing a role in a frame

Table 3: Notation summary

4 A meta-model for role assignment: Confusability

The experiment of the previous section has shown a considerable variance in system performance across roles. The aim of this section is to develop a meta-model which can explain this variance.

The models we have explored in Exp. 1 rely mainly on syntactic features: Even in the combined syntax-semantics model, 24 of the 31 features describe syntactic structure. This predominance of syntactic features can be observed in many current models for semantic role assignment. Accordingly, our meta-model focuses on the uniformity of the mapping from syntactic structure to semantic roles. We formalize the variance in this mapping by the *confusability* of a semantic role. It implements the following hypothesis:

(1) The semantic role assignment systems we study approximate role assignment through grammatical functions.

(2) System performance for a given role depends on the role's *confusability*: A role is highly confusable if the grammatical functions that instantiate it often also instantiate other roles.

By using the ideal, manually assigned grammatical functions that are available from the FrameNet data – and which are not passed on to the learner – our meta-model abstracts over concrete feature sets.

Our definition of confusability proceeds in two steps. First we model the informativity of a grammatical function by the entropy of semantic roles that it maps to. Then we compute the confusability of a role as a weighted average of the entropies of the grammatical functions that realize it.

Grammatical function entropy. Viewing a grammatical function as a random variable with semantic roles as values, we define the entropy of a grammatical function *gf* within the frame *fr* as

$$H_{fr}(gf) = \sum_{fe \in fes(fr)} -p(fe|gf) \log p(fe|gf)$$

where $p(fe|gf) = \frac{f(gf,fe)}{f(gf)}$ is the conditional probability of roles *fe* given *gf* (cf. the notation in Table 3).

Role confusability. The confusability of a role is the sum of its grammatical function entropies, weighted by the conditional probabilities $p(gf|fe) = \frac{f(gf,fe)}{f(fe)}$ of grammatical functions *gf* given *fe*.

$$c_{fr}(fe) = \sum_{gf \in gfs(fr)} p(gf|fe) H_{fr}(gf)$$

Grammatical function entropy					
GF	DEG	THM	DEP	LOC	H
`Mod`	69	43	24	0	1.46
`Comp`	18	491	12	41	0.72
`Ext`	0	17	0	561	0.16
`Head`	0	0	0	273	0.0
`Obj`	0	0	0	3	0.0

Role Confusability						
Role	`Mod`	`Comp`	`Ext`	`Head`	`Obj`	`Conf`
DEG	69	18	0	0	0	1.31
THM	43	491	17	0	0	0.76
DEP	24	12	0	0	0	1.22
LOC	0	41	561	273	3	0.16

Table 4: Grammatical function entropy and role confusability for the frame ABUNDANCE

An example. Table 4 shows the grammatical function entropies and role confusabilities for the frame ABUNDANCE, both computed on the training data. The upper part of Table 4 lists the entropies of the grammatical functions `Mod`, `Comp`, `Ext`, `Head` and `Obj`[3] and the counts $f(gf, fe)$ of occurrences of the grammatical functions together with the roles DEGREE (DEG), THEME (THM), DEPICTIVE (DEP) and LOCATION (LOC). The entropy of `Mod`, with similar numbers of occurrences for three different roles, is relatively high, while `Ext` occurs almost exclusively for one role and has a much lower entropy. The lower part of Table 4 shows the confusability for the same set of roles. The confusability of

[3]See Fillmore and Petruck (2003) for a glossary of FrameNet's grammatical functions.

DEGREE is relatively high even though it is mostly realized by `Mod` because `Mod` has a high entropy, i.e. it indicates multiple roles; LOCATION on the other hand is not very confusable even though it occurs frequently as both `Ext` and `Head`, since both grammatical functions indicate this role.

Related work. Our approach is similar to Pado and Boleda (2004) in that they also use the uniformity of linking as an explanation for performance variations in semantic role assignment. However, their analysis is located at the frame level. We examine individual roles, which allows us to derive a simpler and more intuitive formalization of linking uniformity. Also, our model will ultimately lead us to a different conclusion: the uniformity of linking is a good predictor of the performance of role assignment systems, but only for exclusively syntactic models (see Section 5).

5 Experiment 2: Relating confusability and system performance

In this section, we test the validity of our metamodel. We assess whether confusability, defined in Section 4, can explain the variance in role assignment that we have found in Section 3, by testing the correlation between the two variables.

Experimental setup. We use the same data set (Senseval-3) and the same two classifiers (memory-based and maximum entropy classification) as in Exp. 1. To cover a wider range of models and thus increase the validity of our analysis, we split up the *Syn* feature set from Exp. 1 into the four smaller sets described in the upper part of Table 5. We use these sets individually, combined, and together with the lexical features in the *Sem* set. This results in a total of 20 different models (10 for each classifier), for which we computed role-specific F-scores.

In parallel, we estimated the confusability as described in Section 4, with FrameNet's manually assigned grammatical functions as a basis, using only the training portion of our data. We did not smooth, but omitted roles occurring less than 5 times to avoid sparse and thus unreliable data points. Recall that confusability does not vary with the feature set, since its central asset is to abstract over concrete model parameters and feature sets.

Feature set	F_{MBL}	F_{Maxent}
Path0	70.9	71.3
Path	73.3	72.6
Pt	78.8	79.0
Path/Pt	80.8	79.8
Path/Sibling	76.7	76.6
Pt/Sibling	78.8	79.1
Syn	82.2	82.4
Sem	80.3	80.7
Syn/Sem	87.1	87.5

Table 6: Exp. 2: Results for different feature sets

Results. The F-scores for the subdivided *Syn* feature set are shown in the upper part of Table 6, with the complete *Syn* and *Sem* sets and their combination below. There is a clear relationship between features and F-score: additional features are consistently rewarded with higher performance. Interestingly, phrase type information appears to be a better role predictor than path (compare models *Path* and *Pt*). Also, the semantic feature set alone (*Sem*) performs at over 80% F-Score, slightly better any of the individual syntactic feature groups.

The high F-score variance between individual roles which we have shown for the feature sets *Syn* and *Syn/Sem* in Exp. 1 generalizes to the other feature sets; all individual syntactic feature sets exhibit a higher variance than *Syn*, and *Sem* shows a higher variance than the *Syn/Sem* combination. This does not come as a surprise, since the two models of Exp. 1 use the two richest feature sets, and we would expect less robust behavior for weaker models. Another point to note is that the performance of the two learners is remarkably similar.

The high variance in the F-scores is mirrored in the confusability figures; we obtain an average confusability for our semantic roles of 1.79 with a high standard deviation of 0.84. A scatter plot of F-scores against confusability figures (Fig.1) suggests a linear correlation analysis.

Analysis 1: Correlating confusability and F-score. Since the data does not appear to be normally distributed, we apply Kendall's nonparametric rank test. The results, which are listed in Table 7, show an extremely significant negative correlation between confusability and F-score: higher confus-

Path0	These are features centered around the path from the target lemma to the constituent: the path itself, its length, partial path up to the lowest common ancestor, the grammatical rule that expands the target predicate's parent, relative position of constituent to target
Path	Feature set Path, plus target lemma
Pt	These are features related to phrase type and part of speech: the phrase type of the constituent and its parent, the POS of the constituent first word, last word and head as well as the POS of an informative content word of the constituent (for PP and SBar constituents only: the head of the head's complement), as well as the target lemma
Sibling	Phrase type and POS of the head of the left and right sibling constituent, and the Collins parser's judgment on the argumenthood of the constituent
Syn	This set combines Path, Sibling and Pt. Additional features are: target voice; the constituent's preposition; a feature combining path with target voice and target POS; and two rule-based features judging argumenthood and grammatical function of the constituent
Sem	These are lexical features: Head words of the constituent and of its left and right siblings; leftmost and rightmost word of the constituent; informative content word lemma (see set Pt for details); and the governing verb of the target predicate

Table 5: Feature groups used in the experiments

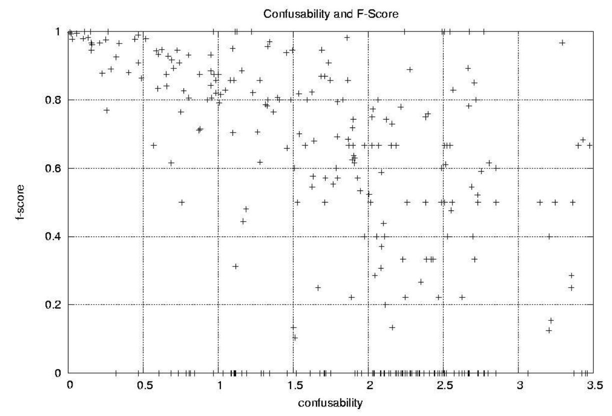


Figure 1: Scatter plot: F-score against confusability (Feature set *Syn*).

	MBL		MaxEnt	
Feature set	z	p	z	p
Path0	-11.72	10^{-15}	-11.76	10^{-15}
Path	-12.29	10^{-15}	-11.23	10^{-15}
Pt	-10.64	10^{-15}	-11.12	10^{-15}
Path/Pt	-11.19	10^{-15}	-10.45	10^{-15}
Path/Sibling	-12.65	10^{-15}	-11.76	10^{-15}
Pt/Sibling	-10.58	10^{-15}	-9.90	10^{-15}
Syn	-9.47	10^{-15}	-9.38	10^{-15}
Sem	-6.90	10^{-11}	-8.23	10^{-15}
Syn/Sem	-8.30	10^{-15}	-8.29	10^{-15}

Table 7: Exp. 2, Analysis 1: Correlation between F-Score and confusability. z: Kendall's tau coefficient, p: significance level

ability appears to be related to lower F-score.

However, note that the correlation is extremely significant even for the model which only uses semantic features. This is unexpected at best and makes a strong interpretation of this correlation doubtful: it is rather likely that there is a third variable with which both F-score and confusability are correlated. The most obvious candidate for such a *confounding variable* is the size of the training set – clearly, we expect our models to perform better with larger training sets. In order to get a more realistic assessment of the relationship between confusability and F-score, we perform an additional analysis to disconfound confusability and frequency.

Analysis 2: Disconfounding confusability and frequency. One way of factoring out the influence of a confounding variable is to perform a partial correlation analysis, which explicitly removes the effects of a third variable when determining the strength of a correlation between two variables. Like a normal correlation analysis, it yields a *partial correlation coefficient*.

Features	MBL		MaxEnt	
	r_c	r_f	r_c	r_f
Path0	-.29***	-.03	-.29***	-.03
Path	-.30***	-.02	-.27***	-.07**
Pt	-.19***	-.11**	-.21***	-.12**
Path/Pt	-.22***	-.07*	-.19***	-.16***
Path/Sibl	-.31***	+.01	-.28***	-.06*
Pt/Sibl	-.20***	-.10**	-.18***	-.16***
Syn	-.10*	-.17***	-.12*	-.19***
Sem	+.01	-.27***	-.02	-.24***
Syn/Sem	+.02	-.25***	-.01	-.25***

Table 8: Exp. 2, Analysis 2: Partial correlation coefficients. r_c: correlation between F-score and confusability, controlling for training set size. r_f: correlation between F-score and training set size, controlling for confusability. Significance levels: ***: $p<0.001$; **: $p<0.01$; *: $p<0.05$.

We first compute partial correlation coefficients between F-score and confusability, controlling for training set size. The results, which indicate the "true" relationship between performance and confusability, are shown in the r_c columns of Table 8. For both learners, confusability is significantly correlated with F-score for all syntactic feature sets, but not for the semantic feature set and for the combined set *Syn/Sem*.

We also compute the partial correlation coefficients between F-score and training set size, controlling for confusability. These figures are reported in the r_f columns of Table 8 and show the "true" relationship between performance and training set size. There is no significant correlation between training set size and performance for simple syntax based-models, but the correlation is highly significant for complex syntactic models and all semantic models.

Discussion. The partial correlation analysis confirms that confusability is a meta-model that can explain the performance of a range of different models for semantic role assignment, namely those models which rely exclusively on syntactic features. Since we used the gold standard features provided by FrameNet and did not introduce implementation- or feature-specific knowledge, this points to a general limitation of syntax-based models. In contrast, semantic features behave completely differently; their contribution is not limited by a role's confusability. At the very least, it cannot be captured by our current meta-model, but the absolute increase in performance indicates that integrating semantics is the way forward, which is surprising given that the purely lexical features we use the present study are usually extremely sparse.

The analysis of the partial correlation between F-score and training set size also allows interesting conclusions. The correlation is not significant for small syntactic feature sets like *Path*, indicating that models for such features can be learned satisfactorily from relatively small training sets (but which are also limited in expressivity). This is markedly different for richer feature sets. Arguably, these feature sets are sparser and can therefore profit more from an increased amount of training data. Again, the effect is most pronounced for the semantic feature set.

6 Conclusion

In this paper, we have formulated a *meta-model* for semantic role assignment. We have used the *confusability* of roles to predict classification performance independently of the classification framework and feature sets used. We have defined role confusability in two steps: First, we have formalized the certainty with which we can predict a semantic role from a given grammatical function with *grammatical function entropy*. Then, we have defined the *confusability of a role* as a weighted sum of grammatical function entropies.

We have found that role confusability is highly significantly correlated with system performance for models based solely on syntactic features. We conclude that syntactic features approximate a description of grammatical functions, but that semantic features model a different aspect of the world.

Much of current research in semantic role assignment is centered on the refinement of syntactic features. Our study suggests that it may be worthwhile to explore the refinement of semantic features as well. The most obvious choice is to investigate features related to selectional preferences. Possible features include goodness of fit relative to pre-computed preferences (Baldewein et al., 2004), named entities (Pradhan et al., 2004), or broad ontological classes like "animate" or "artifact". Following up on this idea, a natural continuation of the present study would be to create a meta-model that subsumes *semantic* features. Such a model could use optimal selectional restrictions as a predictor. The next step would then be to construct a combined meta-model that describes the behavior of systems with both syntactic and semantic features.

Another interesting research direction that our study suggests is the combination of syntactic and semantic models in co-training. Co-training can be sensibly applied only when conditional independence holds for the two target functions and the distribution (Blum and Mitchell, 1998), i.e. when it uses two independent views on the instance set. By pointing out a highly significant distinction between syntactic and semantic features with respect to role confusability, our study provides empirical evidence that syntactic and semantic features model different aspects of the role assignment mapping, and that co-training may be feasible by using syntactic and semantic features as views.

Acknowledgments. We are grateful to the Deutsche Forschungsgemeinschaft (DFG) for funding the SALSA-II project (grant PI-154/9-2).

References

U. Baldewein, K. Erk, S. Pado, D. Prescher. 2004. Semantic role labelling with similarity-based generalisation using em-based clustering. In *Proceedings of SENSEVAL-3*.

A. Blum, T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers.*

H. C. Boas. 2002. Bilingual framenet dictionaries for machine translation. In *Proceedings of LREC 2002*, 1364–1371, Las Palmas, Canary Islands.

L. Burnard, 1995. *User's guide for the British National Corpus*. British National Corpus Consortium, Oxford University Computing Services, 1995.

X. Carreras, L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: semantic role labeling. In *Proceedings of CoNLL 2004*, Boston, MA.

X. Carreras, L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of CoNLL 2005*, Ann Arbor, MI.

M. J. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In A. Joshi, M. Palmer, eds., *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, 184–191, San Francisco. Morgan Kaufmann Publishers.

W. Daelemans, J. Zavrel, K. van der Sloot, A. van den Bosch. 2003. Timbl: Tilburg memory based learner, version 5.0, reference guide. Technical Report ILK 03-10, Tilburg University, 2003. Available from `http://ilk.uvt.nl/downloads/pub/papers/ilk0310.ps.gz`.

C. J. Fillmore, M. R. Petruck. 2003. FrameNet glossary. *International Journal of Lexicography*, 16:359–361.

C. J. Fillmore, C. R. Johnson, M. R. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

C. J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica*, IV(2).

M. Fleischmann, E. Hovy. 2003. A maximum entropy approach to framenet tagging. In *Proceedings of HLT/NAACL 2003*, Edmonton, Canada.

D. Gildea, D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

K. Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In R. Mihalcea, P. Edmonds, eds., *Proceedings of Senseval-3: The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL 2002*, Taipei, Taiwan.

A. Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *Proceedings of the ACL 2004*, Barcelona, Spain.

S. Narayanan, S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of COLING 2004*, Geneva, Switzerland.

S. Pado, G. Boleda Torrent. 2004. The influence of argument structure on semantic role assignment. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin, D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL 2004*, Boston, MA.

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of NeMLaP 1994*.

M. Surdeanu, S. Harabagiu, J. Williams, P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*, Sapporo, Japan.

N. Xue, M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

Improving Statistical MT through Morphological Analysis

Sharon Goldwater
Dept. of Cognitive and Linguistic Sciences
Brown University
sharon_goldwater@brown.edu

David McClosky
Dept. of Computer Science
Brown University
dmcc@cs.brown.edu

Abstract

In statistical machine translation, estimating word-to-word alignment probabilities for the translation model can be difficult due to the problem of sparse data: most words in a given corpus occur at most a handful of times. With a highly inflected language such as Czech, this problem can be particularly severe. In addition, much of the morphological variation seen in Czech words is not reflected in either the morphology or syntax of a language like English. In this work, we show that using morphological analysis to modify the Czech input can improve a Czech-English machine translation system. We investigate several different methods of incorporating morphological information, and show that a system that combines these methods yields the best results. Our final system achieves a BLEU score of .333, as compared to .270 for the baseline word-to-word system.

1 Introduction

In a statistical machine translation task, the goal is to find the most probable translation of some foreign language text f into the desired language e. That is, the system seeks to maximize $P(e|f)$. Rather than maximizing $P(e|f)$ directly, the standard noisy channel approach to translation uses Bayes inversion to split the problem into two separate parts:

$$\underset{e}{\operatorname{argmax}} \mathrm{P}(e|f) = \underset{e}{\operatorname{argmax}} \mathrm{P}(e)\mathrm{P}(f|e) \qquad (1)$$

where $P(e)$ is known as the *language model* and $P(f|e)$ is known as the *translation model*. The limiting factor in machine translation is usually the quality of the translation model, since the monolingual resources needed for training the language model are generally more available than the parallel corpora needed for training the translation model.

Due to the difficulty in obtaining large parallel corpora, sparse data is a serious issue when estimating the parameters of the translation model. This problem is compounded when one or both of the languages involved is a highly inflected language. In this paper, we present a series of experiments suggesting that morphological analysis can be used to reduce data sparseness and increase similarity between languages, thus improving the quality of machine translation for highly inflected languages. Our work is on a language pair in which the input language (Czech) is highly inflected, and the output language (English) is not. We discuss in Section 5 how our methods might be generalized to pairs where both languages are highly inflected.

The plan of this paper is as follows: In Section 2, we review previous work on using morphological analysis for statistical machine translation. In Section 3, we describe several methods for utilizing morphological information in a statistical translation model. Section 4 presents the results of our experiments using these methods. Sections 5 and 6 discuss the results of our experiments and conclude the paper.

2 Previous Work

Until recently, most machine translation projects involved translating between languages with relatively little morphological structure. Nevertheless, a few research projects have investigated the use of morphology to improve translation quality. Niessen and Ney (2000; 2004) report work on German-English translation, where they investigate various types of morphosyntactic restructuring, including merging German verbs with their detached prefixes, annotating a handful of frequent ambiguous German words with POS tags, combining idiomatic multi-word expressions into single words, and undoing question inversion and *do*-insertion in both German and English. In addition, Niessen and Ney (2004) decompose German words into a hierarchical representation using lemmas and morphological tags, and use a MaxEnt model to combine the different levels of representation in the translation model. The results from these papers indicate that on corpus sizes up to 60,000 parallel sentences, the restructuring operations yielded a large improvement in translation quality, but the morphological decomposition provided only a slight additional benefit. However, since German is not as morphologically complex as Czech, we might expect a larger benefit from morphological analysis in Czech.

Another project utilizing morphological analysis for statistical machine translation is described by Lee (2004). Lee's system for Arabic-English translation takes as input POS-tagged English and Arabic text, where the Arabic words have been pre-segmented into stems and affixes. The system performs an initial alignment of the Arabic morphemes to the English words. Based on the consistency of the English POS tag that each Arabic morpheme aligns to, the system determines whether to keep that morpheme as a separate item, merge it back onto the stem, or delete it altogether. In addition, multiple occurrences of the determiner *Al* within a single Arabic noun phrase are deleted (i.e. only one occurrence is allowed). Using a phrase-based translation model, Lee found that *Al*-deletion was more helpful than the rest of the morphological analysis. Also, *Al*-deletion helped for training corpora up to 3.3 million sentences, but the other morphological analysis helped only on the smaller corpus sizes (up to 350,000 parallel sentences). This result is consistent with anecdotal evidence suggesting that morphological analysis becomes less helpful as corpus sizes increase. However, since parallel corpora of hundreds of thousands of sentences or more are often difficult to obtain, it would still be worthwhile to develop a method for improving systems trained on smaller corpora.

Previous results on Czech-English machine translation suggest that morphological analysis may be quite productive for this highly inflected language where there is only a small amount of closely translated material. Čmejrek et al. (2003), while not focusing on the use of morphology, give results indicating that lemmatization of the Czech input improves BLEU score relative to baseline. These results support the earlier findings of Al-Onaizan et al. (1999), who used subjective scoring measures. Al-Onaizan et al. measured translation accuracy not only for lemmatized input, but for an input form they refer to as *Czech'*. Czech' is intended to capture many of the morphological distinctions of English, while discarding those distinctions that are Czech-specific. The Czech' input was created by distinguishing the Czech lemmas for singular and plural nouns, different verb tenses, and various inflections on pronouns. Artificial words were also added automatically in cases where syntactic information in the Czech parse trees indicated that articles, pronouns, or prepositions might be expected in English. The transformation to Czech' provided a small additional increase in translation quality over basic lemmatization.

The experiments described here are similar to those performed by Al-Onaizan et al. (1999), but there are several important differences. First, we use no syntactic analysis of the Czech input. Our intent is to determine how much can be gained by a purely morphological approach to translation. Second, we present some experiments in which we modify the translation model itself to take advantage of morphological information, rather than simply transforming the input. Finally, our use of BLEU scores rather than subjective measurements allows us to perform more detailed evaluation. We examine the effects of each type of morphological information separately.

3 Morphology for MT

Morphological variations in Czech are reflected in several different ways in English. In some cases, such as verb past tenses or noun plurals, morphological distinctions found in Czech are also found in English. In other instances, English may use function words to express a meaning that occurs as a morphological variant in Czech. For example, genitive case marking can often be translated as *of* and instrumental case as *by* or *with*. In still other instances, morphological distinctions made in Czech are either completely absent in English (e.g. gender on common nouns) or are reflected in English syntax (e.g. many case markings). Handling these correspondences between morphology and syntax requires analysis above the lexical level and is therefore beyond the scope of this paper. However, morphological analysis of the Czech input can potentially be used to improve the translation model by exploiting the other types of correspondences we have mentioned.

Before we describe how this can be done, it is important to clarify the kind of morphological analysis we assume in our input. Our data comes from the Prague Czech-English Dependency Treebank (PCEDT) (Hajič, 1998; Čmejrek et al., 2004), the Czech portion of which has been fully annotated with morphological information. Each Czech word in the corpus is associated with an analysis containing the word's lemma and a sequence of morphological

```
Pro/pro/RR--4----------
někoho/někdo/PZM-4----------
by/být/Vc-X---3-------
její/jeho/PSZS1FS3-------
provedení/provedení/NNNS4-----A----
mělo/mít/VpNS---XR-AA---
smysl/smysl/NNIS4-----A----
././Z:-------------
```

Figure 1: A sentence from the PCEDT corpus. Each token is followed by its lemma and a string giving the values of up to 15 morphological tags. Dashes indicates tags that are not applicable for a particular token. This sentence corresponds to the English sentence *It would make sense for somebody to do it.*

tags. These tags provide values along several morphological dimensions, such as part of speech, gender, number, tense, and negation. There are a total of 15 dimensions along which words may be characterized, although most words have a number of dimensions unspecified. An example sentence from the Czech corpus is shown in Figure 1.

In what follows, we describe four different ways that the Czech lemma and tag information can be used to modify the parameters of the translation model. The first three of these are similar to the work of Al-Onaizan et al. (1999) and involve transformations to the input data only. The assumptions underlying the word alignment model $P(f_j|e_i)$ (where f_j and e_i are individual words in an aligned sentence pair) are maintained. The fourth method of incorporating morphological information is novel and changes the alignment model itself.

3.1 Lemmas

A very simple way to modify the input data using morphological information is by replacing each wordform with its associated lemma (see Figure 2). Based on previous results (Al-Onaizan et al., 1999; Čmejrek et al., 2003), we expected that this transformation would lead to an improvement in translation quality due to reduction of data sparseness. However, since lemmatization does remove some useful information from the Czech wordforms, we also tried two alternative lemmatization schemes. First, we tried lemmatizing only certain parts of speech, leaving other parts of speech alone. We reasoned that nouns, verbs, and pronouns all carry inflectional morphology in English, so by lemmatizing only the other parts of speech, we might retain some of the benefits of full lemmatization without losing as much information. We also tried lemmatizing all parts of speech except pronouns, which are very common and therefore should be less affected by sparse data problems.

As a second alternative to full lemmatization, we experimented with lemmatizing only the less frequent wordforms in the corpus. This allows the translation system to use the full wordform information from more frequent forms, where sparse data is less of a problem.

To determine whether knowledge of lemmas was actually necessary, we compared lemmatization with word truncation. We truncated each wordform in the data after a fixed number of characters, as suggested by Och (1995).

3.2 Pseudowords

As discussed earlier, much of the information encoded in Czech morphology is encoded as function words in English. One way to reintroduce some of the information lost during Czech lemmatization is by using some of the morphological tags to add extra "words" to the Czech input. In many cases, these pseudowords will also increase the correspondence of English function words to items in the Czech input. In our system, each pseudoword encodes a single morphological tag (feature/value pair), such as PER_1 ('first person') or TEN_F ('future tense'). Figure 2 shows a Czech input sentence after generating pseudowords for the person feature on verbs.

We expected that the class of tags most likely to be useful as pseudowords would be the person tags, because Czech is a pro-drop language. Using the person tags as pseudowords should simulate the existence of pronouns for the English pronouns to align to. We also expected that negation (which is expressed on verbs in Czech) would be a useful pseudoword, and that case markings might also be helpful since they sometimes correspond to prepositions in English, such as *of*, *with*, or *to*.

3.3 Modified Lemmas

In some cases, such as the past tense, Czech morphology is likely to correspond not to a function word in English, but rather to English inflectional morphology. In order to capture this kind of phenomenon, we experimented with concatenating the Czech morphological tags onto their lemmas instead of inserting them as separate input tokens. See Figure 2 for an example. This concatenation creates distinctions between some lemmas, which will ideally correspond to morphological distinctions made in English. Although this transformation splits the Czech data (relative to pure lemmatization), it still suppresses many of the distinctions made in the full Czech wordforms. We expected that number mark-

Words:	`Pro někoho by její provedení mělo smysl .`
Lemmas:	`pro někdo být jeho provedení mít smysl .`
Lemmas+Pseudowords:	`pro někdo být PER_3 jeho provedení mít PER_X smysl .`
Modified Lemmas:	`pro někdo být+PER_3 jeho provedení mít+PER_X smysl .`

Figure 2: Various transformations of the Czech sentence from Figure 1. The pseudowords and modified lemmas encode the verb person feature, with the values 3 (third person) and X ("any" person).

ing on nouns and tense marking on verbs would be the tags best treated in this way.

3.4 Morphemes

Our final set of experiments used the same input format as the Modified Lemma experiments. However, in this set of experiments, we changed the model used to calculate the word-to-word alignment probabilities. In the standard system, the alignment model parameters $P(f_j|e_i)$ are found using maximum likelihood estimation based on the expected number of times f_j aligns to e_i in the parallel corpus. Our new model assumes a compositional structure for f_j, so that $f_j = f_{j0} \dots f_{jK}$, where f_{j0} is the lemma of f_j, and $f_{j1} \dots f_{jK}$ are morphemes generated from the tags associated with f_j. We assume that every word contains exactly K morphemes, and that the kth morpheme in each word is used to encode the value for the kth class of morphological tag, where the classes (e.g. person or tense) are assigned an ordering beforehand. f_{jk} is assigned a null value if the value of the kth tag class is unspecified for f_j.

Given this decomposition of words into morphemes, and a generative model in which each morpheme in f_j is generated independently conditioned on e_i, we have

$$\mathrm{P}(f_j|e_i) = \prod_{k=0}^{K} \mathrm{P}(f_{jk}|e_i) \tag{2}$$

We can now estimate $\mathrm{P}(f_j|e_i)$ using a slightly modified version of the standard EM algorithm for learning alignment probabilities. During the E step, we calculate the expected alignment counts between Czech morphemes and English words based on the current word alignments, and revise our estimate of $\mathrm{P}(f_j|e_i)$ using Equation 2. The M step of the algorithm remains the same.

The morpheme-based model in Equation 2 is similar to the modified lemma model in that it removes much of the differentiation between Czech wordforms, but leaves the differences that are most likely to appear as inflection on English words. However, it also performs an additional smoothing function. The model assumes that, in the absence of other information, an English word that has aligned mostly to Czech words with a particular morphological tag is more likely to align to another word with this tag than to a Czech word with a different tag. For example, an English word aligned to mostly past tense forms is more likely to align to another past tense form than to a present or future tense form.

4 Experiments

In order to evaluate the effectiveness of the techniques described in the previous section, we ran a number of experiments using data from the PCEDT corpus. The English portion of this corpus (used to train the language model) contains the same material as the Penn WSJ corpus, but with a different division into training, development, and test sets. About 250 sentences each for development and test were translated once into Czech and then back into English by five different translators. These translations are used to calculate BLEU scores. The remainder of the corpus (about 50,000 sentences) is used for training. About 21,000 of the training sentences have been translated into Czech and morphologically annotated for use as a parallel corpus.

Some statistics on the parallel corpus are shown in the graph in Figure 3. This graph illustrates the sparse data problem in Czech that our morphological analysis is intended to address. Although the number of infrequently occurring lemmas is about the same in both English and Czech, the number of infrequently occurring inflected wordforms is approximately twice as high in Czech.[1]

For all of our experiments, we used the same language model, trained with the CMU Statistical Language Modelling Toolkit (Clarkson and Rosenfeld, 1997). Our translation models were trained using GIZA++ (Och and Ney, 2003), which we modi-

[1]Although we did not use it for the experiments in this paper, the PCEDT corpus does contain lemma information for the English data. There is a slight discrepancy between the English and Czech data in the lemma information for pronouns, in that English pronouns (including accusitive, possessive, and other forms) are assigned themselves as lemmas, whereas Czech pronouns are reduced to uninflected forms. Given that pronouns generally have many tokens, this discrepancy should not affect the data in Figure 3.

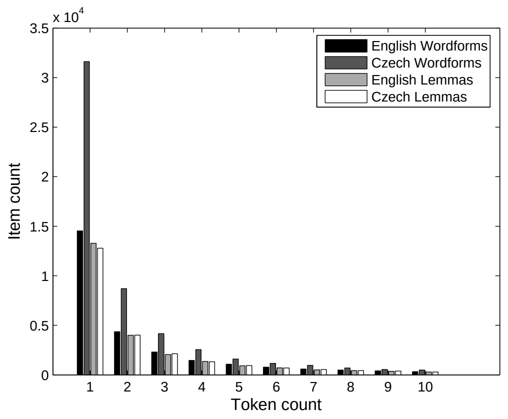


Figure 3: The number of items (full wordforms or lemmas) y appearing in the parallel corpus with a token count of x.

fied as necessary for the morpheme-based experiments. We used the ISI ReWrite Decoder (Marcu and Germann, 2005) for producing translations. Before beginning our experiments, we obtained a baseline BLEU score by training a standard word-to-word translation model. Our baseline results indicate that the test set for this corpus is considerably more difficult than the development set: word-to-word scores were .311 (development) and .270 (test).

4.1 Lemmas

As Figure 3 shows, lemmatization of the Czech corpus cuts the number of unique items by more than half, and the number of items with no more than ten occurrences by nearly half. The lemmatization BLEU scores in Table 1 indicate that this has a large impact on the quality of translation. As expected, full lemmatization performed better than word-to-word translation, with an an improvement of about .04 in the development set BLEU score and .03 in the test set. (In this and the following experiments, BLEU score differences of .009 or more are significant at the .05 level.) Experiments on the development set showed that leaving certain parts of speech unlemmatized did not improve results, but lemmatizing only low-frequency words did. A frequency cutoff of 50 worked best on the development set (i.e. only words with frequency less than 50 were lemmatized). Despite the improvement on the development set, using this cutoff with the test set yielded only a non-significant improvement over full lemmatization.

The results of these lemmatization experiments support the argument that lemmatization improves translation quality by reducing data sparseness, but also removes potentially useful information. Our results suggest that lemmatizing only infrequent words may, in some cases, work better than lemmatizing all words.

	Dev	Test
word-to-word	.311	.270
lemmatize all	.355	.299
except Pro	.350	
except Pro, V, N	.346	
lemmatize $n < 50$	.370	.306
truncate all	.353	.283

Table 1: BLEU scores for the word-to-word baseline, lemmatization, and word truncation experiments.

As Table 1 indicates, it is possible to get some of the benefits of lemmatization without using any morphological knowledge at all. For both dev and test sets, truncating words to 6 characters (the best length on the dev set) provided a significant improvement over word-to-word translation, but was also significantly worse than the best lemmatization scores. Changing the frequency cutoff for truncation did not produce any significant differences in the BLEU score.

4.2 Pseudowords

Results for the pseudoword experiments on the development set are shown in the first column of Table 2. Note that in these (and the following) experiments, we treated all words the same way regardless of their frequency, so the effects of adding morphological information are in comparison to the full lemmatization scheme. In most of our experiments, we added morphological information for only a single class of tags at a time in order to determine the effects of each class individually. The classes we used were verb person (PER), verb tense (TEN), noun number (NUM), noun case (CASE), and negation (NEG).

Most of the results of the pseudoword experiments confirm our expectations. Adding the verb person tags was helpful, and examination of the alignments revealed that they did indeed align to English pronouns with high probability. The noun number tags did not help, since plurality is expressed as an affix in English. Negation tags helped slightly, though the improvement was not significant. This is probably because negation tags are relatively infrequent, as can be seen in Table 3. The addition of pseudowords for case did not yield an improvement, probably because these pseudowords were so frequent. The additional ambiguity caused by so many extra words likely overwhelmed any positive effect.

A somewhat puzzling result is the behavior of the

Tag type	Pseudo	Mod-Lem	Morph
PER	.365	.356	.356
TEN	.365	.361	.364
PER,TEN	.355	.362	.355
NUM	.354	.367	.361
CASE	.353	.340	.337
NEG	.357	.356	.353

Table 2: BLEU scores indicating the results of incorporating the information from different classes of morphological tags in the the experiments using pseudowords (Pseudo), modified lemmas (Mod-Lem), and morphemes (Morph). Scores are from the development set. Differences of .009 are significant ($p < .05$).

Tag class	Count	Avg/sentence
PER	49700	2.35
TEN	47744	2.26
past	22544	1.07
pres	20291	0.96
fut	1707	0.08
'any'	3202	0.15
NUM	151646	7.17
CASE	151646	7.17
NEG	3326	0.16

Table 3: Number of occurrences of each class of tags in the Czech training data.

verb tense tags. With the exception of future tense, English generally does not mark tense with an auxiliary. Yet Table 3 shows that only a very small percentage of sentences have a future tense marker, so it seems unlikely that this explains the positive effects of the tense pseudowords. In fact, we tried adding only future tense pseudowords to the lemmatized Czech data, and found that the results were no better than basic lemmatization.

The other unusual behavior we see with pseudowords is that when verb person and tense tags are combined, they seem to cancel each other out, resulting in a score that is no better than lemmatization alone. Examination of the alignments did not reveal any obvious reason for this effect.

4.3 Modified Lemmas

As shown in the second column of Table 2, the number and tense tags yield an improvement under the modified lemma transformation, while the person tags do not. Again, this confirms our predictions based on the morphology of English.

Our results using the case tags under this model actually decreased performance, but this is not surprising given that differentiating Czech lemmas based on case marking creates as much as a 7-way split of the data (there are seven cases in Czech), without adding much information that would be useful in English.

4.4 Morphemes

BLEU scores for the morpheme-based model are given in the third column of Table 2. None of the differences in scores between this model and the modified lemma model are significant, although the trend for most of the tag classes is for this model to perform slightly worse. This suggests that the type of smoothing induced by the morpheme-based model may not be as helpful as simply attempting to create Czech words that reflect the same morphological distinctions as the English words. In Section 5, we propose a generalized version of the morpheme model that might be an improvement.

4.5 Combined Model

In the experiments described so far, we used only a single method at a time of incorporating morphological information into the translation process. However, it is straightforward to combine the pseudoword method with either the modified-lemma or morpheme-based methods by using pseudowords for certain tags and attaching others to the Czech lemmas. The experiments described above allowed us to confirm our intuitions about how each class of tags should be treated under such a combined model. We then created a model using the pseudoword treatment of the person and negation tags, and the modified lemma treatment of number and tense. We did not use the case tags in this model, since they did not seem to yield an improvement in any of the three basic morphological models.

Our combined model achieved a BLEU score of .390 (development) and .333 (test), outperforming the models in all of our previous experiments.

5 Discussion

The results of our experiments provide additional support for the findings of previous researchers that using morphological analysis can improve the quality of statistical machine translation for highly-inflected languages. While human judgment is probably the best metric for evaluating translation quality, our use of the automatically-derived BLEU score allowed us to easily compare many different translation models and evaluate the effects of each one individually. We found that simple lemmatization, by significantly reducing the sparse data problem, was quite effective despite the loss of information involved. Lemmatizing the less frequent words in the corpus seemed to increase performance slightly, but these results were inconclusive. Word truncation, which requires no morphological information at all, was effective at increasing scores over the word-to-word baseline, but did not perform quite as well as lemmatization. This result conflicts with Och's (Och, 1995), and is likely due to the much smaller size of our corpus. In any case, our results suggest that lemmatization or word truncation could yield a significant improvement in the quality of translation from a highly-inflected to a less-inflected language, even when limited morphological information is available.

Our primary results concern the use of full morphological information. We found that certain tags were more useful when we treated them as discrete input words, while others provided a greater benefit when attached directly to their lemmas. The best choice of which method to use for each class of tags seems to correspond closely with how that class of information is expressed in English (either using function words or inflection). In a sense, the goal of the morphological analysis is to make the Czech input data more English-like by suppressing unnecessary morphological distinctions and expressing necessary distinctions in ways that are similar to English. This sort of procedure could be taken further by incorporating syntactic information as well, but as we stated earlier, our goal was to determine exactly how much benefit we could derive from a strictly morphological approach.

In the work we have presented, the output language (English) is low in inflection. We therefore considered it less important to perform morphological analysis on the English data. However, we expect that the work described here could be generalized to highly inflected output languages by doing morphological analysis on both the input and output languages. The most promising way to do this seems to be by extending the morpheme-based translation model in Equation 2 to incorporate morphemes in both languages, so that

$$\mathrm{P}(f_j|e_i) = \prod_{k=0}^{K} \mathrm{P}(f_{jk}|e_{ik}) \tag{3}$$

where f_{jk} are the morphemes in the input language, and e_{ik} are the corresponding morphemes in the output language. This extended model may also prove a benefit to Czech-English translation; we are currently investigating this possibility.

In this work, we used a word-based translation system due to the availability of source code that could be modified for our morph experiments. An obvious extension to the current work would be to move to a phrase-based translation system. One advantage of phrase-based models is their ability to align phrases in one language to morphologically complex words in the other language. However, this feature still suffers from the same sparse data problems as a word-based system: if a morphologically complex word only appears a handful of times in the training corpus, the system will have difficulty determining its (phrasal or word) alignment. We expect that morphological analysis would still be helpful in this situation, at the very least because it can be used to remove distinctions that appear in only one language.

6 Conclusion

In this paper we used morphological analysis of Czech to improve a Czech-English statistical machine translation system. We have argued that this improvement was primarily due to a reduction of the sparse data problem caused by the highly inflected nature of Czech. An alternative method for reducing sparse data is to use a larger parallel corpus; however, it is often easier to obtain additional monolingual resources, such as a morphological analyzer or tagged corpus, than additional parallel data for a specific language pair. For that reason, we believe that the approach taken here is a promising one.

We have described several different ways of using morphological information for machine translation, and have shown how these can be combined to yield an improved translation model. In general, we would not expect the exact combination of techniques that yielded our best results for Czech-English to be optimal for other language pairs. Rather, we have suggested that these techniques should be combined in a way that makes the input language more similar to the output language. Although this combination will need to be determined for each language pair, the general approach outlined here should provide benefits for any MT system involving a highly inflected language.

Acknowledgements

We would like to thank Eugene Charniak and the members of BLLIP for their encouragement and helpful suggestions. This research was partially supported by NSF awards IGERT 9870676 and ITR 0085940.

References

Y. Al-Onaizan, J. Cuřin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky. 1999. Statistical machine translation. Final Report, JHU Summer Workshop 1999.

P. Clarkson and R. Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge Toolkit. In *Proceedings of ESCA Eurospeech*. Current version available at http://mi.eng.cam.ac.uk/∼prc14/toolkit.html.

J. Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press.

Y. Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings NAACL*.

D. Marcu and U. Germann. 2005. The ISI ReWrite Decoder 1.0.0a. Available at http://www.isi.edu/licensed-sw/rewrite-decoder/.

S. Niessen and H. Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *Proceedings of COLING*.

S. Niessen and H. Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic analysis. *Computational Linguistics*, 30(2):181–204.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

F. Och. 1995. Statistical machine translation: The fabulous present and future. Invited talk at the Workshop on Building and Using Parallel Texts at ACL'05.

M. Čmejrek, J. Cuřín, and J. Havelka. 2003. Czech-english dependency-based machine translation. In *Proceedings of EACL*.

M. Čmejrek, J. Cuřín, J. Havelka, J. Hajič, and V. Kuboň. 2004. Prague czech-english dependecy treebank: Syntactically annotated resources for machine translation. In *4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal.

A Translation Model for Sentence Retrieval

Vanessa Murdock and **W. Bruce Croft**
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003
{vanessa,croft}@cs.umass.edu

Abstract

In this work we propose a translation model for monolingual sentence retrieval. We propose four methods for constructing a parallel corpus. Of the four methods proposed, a lexicon learned from a bilingual Arabic-English corpus aligned at the sentence level performs best, significantly improving results over the query likelihood baseline. Further, we demonstrate that smoothing from the local context of the sentence improves retrieval over the query likelihood baseline.

1 Introduction

Sentence retrieval is the task of retrieving a relevant sentence in response to a user's query. Tasks such as question answering, novelty detection and summarization often incorporate a sentence retrieval module. In previous work we examined sentence retrieval for question answering (Murdock and Croft, 2004). This involves the comparison of two well-formed sentences, one a question, one a statement. In this work we compare well-formed sentences to queries, which can be typical keyword queries of 1 to 3 terms, or a set of sentences or sentence fragments. The TREC Novelty Track provides this type of data in the form of topic titles and descriptions, and sentence-level relevance judgments for a small subset of the collection.

We present a translation model specifically for monolingual data, and show that it significantly improves sentence retrieval over query-likelihood. Translation models train on a parallel corpus and in previous work we used a corpus of question/answer pairs. No such corpus is available for the novelty data, so in this paper we present four ways to construct a parallel corpus, to estimate a translation model.

Many systems treat sentence retrieval as a type of document or passage retrieval. In our data a sentence is an average of 18 words, most of which occur once. A document is an average of 700 words, many of which are multiples of the same term. It is much less likely for a word and its synonym terms to appear in the same sentence than in the same document.

Passages may be any length, either fixed or variable, but are somewhat arbitrarily designated. Many systems that have a passage retrieval module, on closer inspection have defined the passage to be a sentence. What is needed is a sentence retrieval mechanism that retains the benefits of passage retrieval, where a passage is longer than a sentence. We propose that smoothing from the local context of the sentence improves retrieval over the query likelihood baseline, and the larger the context, the greater the improvement.

We describe our translation model in section 2, along with our smoothing approach. In section 3 we discuss previous work in sentence retrieval for the Novelty task, and translation models for information retrieval tasks. Section 4 presents four ways to estimate a translation model, in the absence of a parallel corpus, and presents our experimental results. We discuss the results in section 5, and present our conclusions and future work in section 6.

2 Methodology

Our data was provided by NIST, as part of the TREC Novelty Track[1]. The documents for the TREC Novelty Track in 2002 were taken from the TREC volumes 4 and 5, and consist of news articles from the Financial Times, the Foreign Broadcast Information Service, and the Los Angeles Times from non-overlapping years. In 2003 and 2004, the documents were taken from the Aquaint Corpus, which is distributed by the Linguistic Data Consortium[2] and consists of newswire text in English from the Xinhua News Service, the New York Times, and the Associated Press from overlapping years.

We retrieved the top 1000 documents for each topic from the TREC and Aquaint collections, and sentence segmented the documents using MXTerminator (Reynar and Ratnaparkhi, 1997), which is a freely available sentence boundary detector. Each topic was indexed separately and had an average of 30,000 sentences. It was impractical to do sentence-level relevance assessments for the complete set of 150,000 documents, so we used the relevance assessments provided as part of the Novelty task, recognizing that the results are a lower bound on performance, because the relevance assessments do not cover the collection. The relevance assessments cover 25 known relevant documents for each topic.

We evaluated precision at N documents because many systems using sentence retrieval emphasize the results at the top of the ranked list, and are less concerned with the overall quality of the list.

2.1 Translation Models

We incorporated a machine translation model in two steps: estimation and ranking. In the estimation step, the probability that a term in the sentence "translates" to a term in the query is estimated using the implementation of IBM Model 1 (Brown et al., 1990) in GIZA++ (Al-Onaizan et al., 1999) out-of-the-box without alteration. In the ranking step we incorporate the translation probabilities into the query-likelihood framework.

In Berger and Lafferty (1999), the IBM Model 1 is incorporated thus:

$$P(q_i|S) = \sum_{j=1}^{m} P(q_i|s_j)P(s_j|S) \tag{1}$$

where $P(q_i|s_j)$ is the probability that term s_j in the sentence translates to term q_i in the query. If the translation probabilities are modified such that $P(q_i|s_j) = 1$ if $q_i = s_j$ and 0 otherwise, this is Berger and Lafferty's "Model 0", and it is exactly the query-likelihood model (described in section 2.2).

A major difference between machine translation and sentence retrieval is that machine translation assumes there is little, if any, overlap in the vocabularies of the two languages. In sentence retrieval we depend heavily on the overlap between the two vocabularies. With the Berger and Lafferty formulation in equation 1, the probability of a word translating to itself is estimated as a fraction of the probability of the word translating to all other words. Because the probabilities must sum to one, if there are any other translations for a given word, its self-translation probability will be less than 1.0. To accommodate this monolingual condition, we make the following improvement.

Let $t_i = 1$ if there exists a term in the sentence s_j such that $q_i = s_j$, and 0 otherwise:

$$\sum_{1 \le j \le n} p(q_i|s_j)p(s_j|S) \Longrightarrow$$
$$t_i p(q_i|S) + (1 - t_i) \sum_{1 \le j \le n, s_j \ne q_i} p(q_i|s_j)p(s_j|S) \tag{2}$$

The translation probabilities still sum to one. We determined empirically that this adjustment improved the results over IBM model 1, and over Berger and Lafferty model 0.

[1]http://trec.nist.gov
[2]http://www.ldc.upenn.edu

2.2 Document Smoothing

Query likelihood is a generative model that assumes that the sentence is a sample of a multinomial distribution of terms. Sentences are ranked according to the probability they generate the query. We estimate this probability by interpolating the term distribution in the sentence with the term distribution in the collection:

$$P(Q|S) = P(S)\prod_{i=1}^{|Q|}\Big(\lambda P(q_i|S) + (1-\lambda)P(q_i|C)\Big) \tag{3}$$

where Q is the query, S is the sentence, $P(S)$ is the (uniform) prior probability of the sentence, $P(q_i|S)$ is the probability that term q_i in the query appears in the sentence, and $P(q_i|C)$ is the probability that q_i appears in the collection.

In the experiments with document smoothing, we estimate the probability of a sentence generating the query:

$$P(Q|S) = \\ P(S)\prod_{i=1}^{|Q|}\Big(\alpha P(q_i|S) + \beta P(q_i|D_S) + \gamma P(q_i|C)\Big) \tag{4}$$

where $\alpha+\beta+\gamma = 1.0$ and $P(q_i|D_S)$ is the probability that the term q_i in the query appears in the document the sentence came from. In our case, since the sentences for each topic are indexed separately, the collection statistics are in reference to the documents in the individual topic index.

3 Previous Work

The TREC Novelty Track ran for three years, from 2002 to 2004. Overviews of the track can be found in (Harman, 2002), (Soboroff and Harman, 2003) and (Soboroff, 2004). A number of systems use traditional information retrieval techniques for sentence retrieval, using various techniques to compensate for the sparse term distributions in sentences. The University of Massachusetts (Larkey et al., 2002) and Carnegie Mellon University (Collins-Thompson et al., 2002) both ranked sentences by the cosine similarity of the sentence vector to the query vector of tf.idf-weighted terms. Amsterdam University (Monz et al., 2002) used tfc.nfx term weighting which is a variant of tf.idf term weighting that normalizes the lengths of the document vectors. Meiji University (Ohgaya et al., 2003) expanded the query with concept groups, and then ranked the sentences by the cosine similarity between the expanded topic vector and the sentence vector.

Berger and Lafferty (1999) proposed the use of translation models for (mono-lingual) document retrieval. They used IBM Model 1 (Brown et al., 1990), to rank documents according to their translation probability, given the query. They make no adjustment for the fact that the query and the document are in the same language, and instead rely on the translation model to learn the appropriate weights for word pairs. The models are trained on parallel data artificially constructed from the mutual information distribution of terms in the document. The results presented either were not tested for statistical significance, or they were not statistically significant, because no significance results were given.

Berger et al. (2000) used IBM Model 1 to rank answers to questions in call-center data. In their data, there were no answers that were not in response to at least one of the questions, and all questions had at least one answer. Furthermore, there are multiples of the same question. The task is to match questions and answers, given that every question has at least one match in the data. The translation models performed better for this task than the tf.idf baseline.

4 Experiments and Results

In this section we describe four methods for estimating a translation model in the absence of a parallel corpus. We describe experimental results for each of the translation models, as well as for document smoothing.

4.1 Mutual Information and TREC

As in Berger and Lafferty (1999), a set of documents was selected at random from the TREC collection, and for each document we

	Query Likelihood	MT (MI)	MT (TREC)
Prec@5	0.1176	0.1149	0.1392*
Prec@10	0.1115	0.1047	0.1095
Prec@15	0.1023	0.0928*	0.0977
Prec@20	0.0973	0.0882*	0.0936
Prec@30	0.0890	0.0865	0.0874
Prec@100	0.0733	0.0680*	0.0705
R-Prec	0.0672	0.0642*	0.0671
Ave Prec	0.0257	0.0258	0.0264

Table 1: Comparing translation model-based retrieval with description queries. "TREC" and "MI" are two ways to estimate a translation model. Results with an asterisk are significant at the .05 level with a two-tailed t-test.

	Event		Opinion	
	Query Lklhd	MT (TREC)	Query Lklhd	MT (TREC)
Prec@5	0.1149	0.1307	0.1234	0.1574
Prec@10	0.1089	0.1079	0.1170	0.1128
Prec@15	0.1036	0.1030	0.0993	0.0865
Prec@20	0.0985	0.0980	0.0947	0.0840
Prec@30	0.0901	0.0894	0.0865	0.0830
Prec@100	0.0729	0.0719	0.0743	0.0674
R-Prec	0.0658	0.0694	0.0701	0.0622
Ave Prec	0.0275	0.0289	0.0219	0.0211

Table 2: Comparing translation-based retrieval for description queries, using the relevance judgments provided by NIST. The translation model was trained from TREC topics.

constructed a distribution according to each term's mutual information with the document, and randomly generated five queries of 8 words according to this distribution. We were retrieving sentences rather than documents, so each sentence in the document was ranked according to its probability of having generated the query, and then the query was aligned with the top 5 sentences. We call this approach "MI".

The second approach uses the TREC topic titles and descriptions aligned with the top 5 retrieved sentences from documents known to be relevant to those topics, excluding topics that were included in the Novelty data. We call this approach "TREC".

Table 1 shows the results of incorporating translations for topic descriptions. Results in the tables with an asterisk are significant at the .05 level using a two-tailed t-test. The results for sentence retrieval are lower than those typically obtained for document retrieval. Manual inspection of the results indicates that the actual precision is much higher, and resembles the results for document retrieval. The lower results are an artifact of the way the relevance assessments were obtained. The sentence-level judgements from the TREC Novelty Track are only for 25 documents per topic.

The Novelty data from 2003-2004 consists of event and opinion queries. We observed that a number of the topic descriptions for event topics had a high degree of vocabulary overlap with the sentences in our data. This was not true of the opinion queries. The results of using a translation-based retrieval on description queries are given in table 2, broken down by the sentiment of the query. The Novelty queries from 2002 were included in the "event" set.

Not all of the sentences judged relevant to opinion topics express opinions. To assess opinion-relevance we evaluated the top 10 sentences, and marked sentences that expressed opinions. In our data approximately 10% of sentences in the top 10 express opinions. Table 3 shows the result of using a translation model trained on TREC data for description queries, broken down by sentiment, with the baselines evaluated for this particular set of relevance judgments. For opinion questions, the column labeled "topical" indicates topical relevance. The column labeled "opinion" indicates topical relevance that also expresses an opinion.

If we consider a sentence relevant to an opinion question only if it expresses an opinion, we see improvement in the results at the top of the ranked list for those queries, using a translation model trained on TREC data. Of the 150 topics, only 50 were opinion topics, so although the magnitude of the improvement in opinion queries is large the results are not statistically

	Topical Rel		Express Opin	
	Query Lklhd	MT (TREC)	Query Lklhd	MT (TREC)
Prec@5	.7289	.7111	.3300	.3900
Prec@10	.7089	.6867	.3125	.3775
Prec@15	.5363	.4919	.2350	.2717
Prec@20	.4300	.4033	.1875	.2188
Prec@30	.3170	.2970	.1408	.1617
Prec@100	.1236	.1131*	.0587	.0580
R-Prec	.4834	.4653	.2947	.3597*
Ave Prec	.4996	.4696	.2563	.3177

Table 3: Comparison of translation retrieval on opinion queries, using truth data we created to evaluate opinion questions. Translation models were trained with TREC data. Results with an asterisk are significant at the .05 level using a two-tailed t-test.

significant with respect to the baseline.

4.2 Lexicons

External lexicons are often useful for translation and query expansion. The most obvious approach was to incorporate a thesaurus into the training process in GIZA as a dictionary, which affects the statistics in the first iteration of EM. This is intended to improve the quality of the alignments over subsequent iterations. We incorporated the thesauri into the training process of the data generated from the artificial mutual information distribution. The dictionaries had almost no effect on the results.

4.2.1 WordNet

We created a parallel corpus of synonym-term pairs from WordNet, and added this data to the artificial mutual information data to train the translation model. The results of using this approach to retrieve sentences using title queries are in figure 1, labeled "MI_WN". Using WordNet alone, without the mutual information data, is labeled "WN_Only". The results are statistically significant using a Wilcoxon sign test at the .05 level for precision at .10, .20 and .60. Query likelihood retrieval is the baseline. The results for description queries are not shown, and were not significantly different from the baseline.

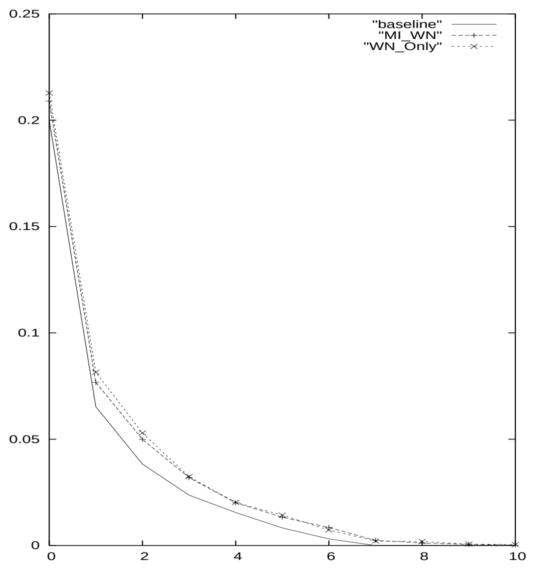


Figure 1: Comparing interpolated recall-precision for title queries using WordNet. The results are statistically significant using a Wilcoxon sign test at the .05 level, for precision at .10, .20 and .60.

4.2.2 Arabic-English corpus

Xu et al. (2002) derive an Arabic thesaurus from a parallel corpus. We derived an English thesaurus using the same approach, from a pair of English-Arabic/Arabic-English lexicons, learned from a parallel corpus. We assumed that if two English terms translate to the same Arabic term, the English terms are synonyms whose probability is given by

$$P(e_2|e_1) = \sum_{a \in A} P(e_2|a)P(a|e_1) \qquad (5)$$

Figure 2 shows the interpolated recall-precision of these results, for description queries. The English terms were not stemmed, and so the baseline query-likelihood results are also not stemmed. The results are statistically significant using a Wilcoxon sign test at the .05 level, for all retrieval levels. Not shown is the average precision, which is also significantly better for the Arabic-English lexicon than for the query-likelihood. The results for title queries are not shown, but are similar to those for descriptions.

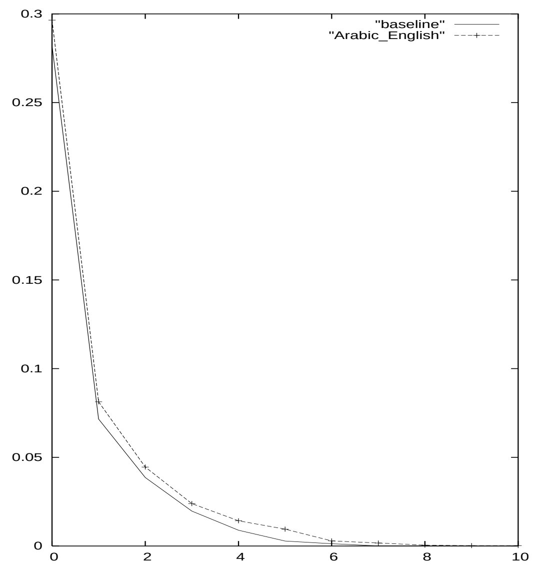


Figure 2: Comparing interpolated recall-precision for description queries using a pair of Arabic-English, English-Arabic lexicons. The results are statistically significant using a Wilcoxon sign test at the .05 level, for precision all recall levels.

4.3 Document Smoothing

Smucker and Allan (2005) demonstrated that under certain conditions, Jelinek-Mercer smoothing is equivalent to Dirichlet smoothing, and that the advantage of Dirichlet smoothing is derived from the fact that it smoothes long documents less than shorter documents. In our data there is much less variance in the length of a sentence than in the length of a document, thus we do not expect to see as great a benefit in performance from Dirichlet smoothing as has been reported in Zhai and Lafferty (2001). In fact we tried Absolute Discounting, Dirichlet, Jelinek-Mercer and Laplace smoothing and found them to produce equivalent results.

The vast majority of sentences in our data are not stand-alone units, and the topic of the sentence is also the topic of surrounding sentences. We took a context of the surrounding 5 sentences, and the surrounding 11 sentences (about 1/3 of the whole document). The sentences were smoothed from the surrounding context, backing-off to the whole document, using Jelinek-Mercer smoothing. Table 4 shows a comparison of the amount of context. Smoothing from the local context is clearly better than the baseline result.

	Query Lklhd	5 Sents	11 Sents
Prec@5	0.1203	0.1527*	0.1541*
Prec@10	0.1122	0.1446*	0.1419*
Prec@15	0.1018	0.1329*	0.1405*
Prec@20	0.0973	0.1311*	0.1345*
Prec@30	0.0890	0.1191*	0.1286*
Prec@100	0.0732	0.0935*	0.1006*
R-Prec	0.0672	0.0881*	0.0933*
Ave Prec	0.0257	0.0410*	0.0485*

Table 4: Comparison of smoothing context on description queries, retrieving sentences from the top 1000 documents. Results with an asterisk are significant at the .05 level using a two-tailed t-test.

We investigated the effect of smoothing from the entire document. Table 5 shows the results. Both topic titles and descriptions get significantly better results with document smoothing.

4.4 Novelty Relevance Task

In the TREC Novelty Track, participants are given a set of 25 documents most of which are relevant for each topic. If we believe that a document is relevant because it has relevant sentences in it, then a "good" sentence would come from a "good" document. This would suggest that smoothing from the document the sentence came from would improve retrieval. We found that for title queries document smoothing improved precision in the top 5 documents by 12.5%, which is statistically significant using a two-tailed t-test at the .05 level. Precision in the top 10 - 100 documents also improved results by an average of 5%, but the result is not statistically significant. For description queries, smoothing from the document had no effect.

For title queries, translation models improve the average precision, and R-Precision. For both title and description queries, the number of relevant documents that are retrieved is also improved with translation models.

	Title		Description	
	Query Lklhd	Doc Smth	Query Lklhd	Doc Smth
Prec@5	.0765	.2268*	.1203	.2362*
Prec@10	.0805	.2262*	.1122	.2128*
Prec@15	.0814	.2192*	.1018	.2000*
Prec@20	.0765	.2124*	.0973	.1893*
Prec@30	.0765	.2007*	.0890	.1743*
Prec@100	.0675	.1638*	.0732	.1335*
R-Prec	.0646	.1379*	.0672	.1226*
Ave Prec	.0243	.0796*	.0257	.0749*

Table 5: Comparison of document smoothing to query likelihood retrieving sentences from the top 1000 documents. Results with an asterisk are significant at the .05 level using a two-tailed t-test.

5 Discussion

The results for sentence retrieval are low, in comparison to results we would expect for document retrieval. We might think that although we show improvements, nothing is working well. In reality, the relevance assessments provided by NIST as part of the Novelty Track only cover 25 documents per topic. Evaluating the top 10 sentences by hand shows that the systems give a performance comparable to document retrieval systems, and the low numbers are the result of a lack of coverage in the assessments. Unfortunately, there is no collection of documents of significant size, where the relevance assessments at the sentence level cover the collection. Constructing such a corpus would be a major undertaking, outside of the scope of this paper.

The best performing method of constructing a parallel corpus used a bilingual lexicon derived from a sentence-aligned Arabic-English parallel corpus. This suggests that data in which sentences are actually translations of one another, as opposed to sentences aligned with key terms from the document, yield a higher quality lexicon. The model trained on the parallel corpus of TREC topics and relevant sentences performed better than the MI corpus, but not as well as the Arabic-English corpus. The TREC corpus consisted of approximately 15,000 sentence pairs, whereas the Arabic-English corpus was trained on more than a million sentence pairs. This may account in part for the higher quality results. In addition, the TREC corpus was created by retrieving the top 5 sentences from each relevant document. Even when the document is known to be relevant, the retrieval process is noisy. Furthermore, although there were 15,000 sentence pairs, there were only 450 unique queries, limiting the size of the source vocabulary.

Opinion topics have much less vocabulary overlap with relevant sentences than do event topics. Translation models would be expected to perform better when retrieving sentences that contain synonym or related terms. For sentences that have exact matches in the query, query likelihood will perform better.

We find that smoothing from the local context of the sentence performs significantly better than the baseline retrieval. The sentences are all about the same length, so there is no performance advantage to using Dirichlet smoothing, whose smoothing parameter is a function of the document length. The smoothing parameters gave very little weight to the collection. As sentences have few terms, relative to documents, matching a term in the query is a good indication of relevance.

6 Conclusions and Future Work

We have shown that translation models improve retrieval for title and opinion queries, and that a translation model derived from a high-quality bilingual lexicon significantly improves retrieval for title and description queries. Smoothing from the local context of a sentence dramatically improves retrieval, with smoothing from the document that contains the sentence performing the best.

We evaluated sentences based on lexical similarity, but structural similarity is also an important measure, which we plan to investigate in the future. The translation model we used was the most basic model. We used this model because it had been shown effective in document retrieval, and was easily incorporated in the query-likelihood framework, but we intend to explore more sophisticated translation models, and better alignment mechanisms. Preliminary results suggest that sentence retrieval can be used to improve document retrieval, but we plan a more extensive investigation of evaluating document similarity and relevance based on sentence-level similarity.

7 Acknowledgements

The authors would like to thank Leah Larkey for her Arabic-English lexicons. This work was supported in part by the Center for Intelligent Information Retrieval, in part by Advanced Research and Development Activity and NSF grant #CCF-0205575 , and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, I. D. Melamed, F. J. Och, D. Purdy, N. A. Smith, and D. Yarowsky. 1999. Statistical machine translation, final report, JHU workshop.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22nd Annual Conference on Research and Development in Information Retrieval (ACM SIGIR)*.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of the 23rd Annual Conference on Research and Development in Information Retrieval (ACM SIGIR)*, pages 192–199.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelineck, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Kevyn Collins-Thompson, Paul Ogilvie, Yi Zhang, and Jamie Callan. 2002. Information filtering, novelty detection and named-page finding. In *Proceedings of the Eleventh Text Retrieval Conference (TREC)*.

Donna Harman. 2002. Overview of the TREC 2002 novelty track. In *Proceedings of the Eleventh Text Retrieval Conference (TREC)*.

Leah Larkey, James Allan, Margie Connell, Alvaro Bolivar, and Courtney Wade. 2002. UMass at TREC 2002: Cross language and novelty tracks. In *Proceedings of the Eleventh Text Retrieval Conference (TREC)*, page 721.

Christof Monz, Jaap Kamps, and Maarten de Rijke. 2002. The University of Amsterdam at TREC 2002. In *Proceedings of the Eleventh Text Retrieval Conference (TREC)*.

Vanessa Murdock and W. Bruce Croft. 2004. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of the Information Retrieval for Question Answering Workshop at SIGIR 2004*.

Ryosuke Ohgaya, Akiyoshi Shimmura, and Tomohiro Takagi. 2003. Meiji University web and novelty track experiments at TREC 2003. In *Proceedings of the Twelth Text Retrieval Conference (TREC)*.

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 (ANLP)*. http://www.cis.upenn.edu/ãdwait/statnlp.html.

Mark Smucker and James Allan. 2005. An investigation of dirichlet prior smoothing's performance advantage. Technical Report IR-391, The University of Massachusetts, The Center for Intelligent Information Retrieval.

Ian Soboroff and Donna Harman. 2003. Overview of the TREC 2003 novelty track. In *Proceedings of the Twelfth Text Retrieval Conference (TREC)*.

Ian Soboroff. 2004. Overview of the TREC 2004 novelty track. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC)*. forthcoming.

Jinxi Xu, Alexander Fraser, and Ralph Weischedel. 2002. Empirical studies in strategies for arabic retrieval. In *Proceedings of the 25th Annual Conference on Research and Development in Information Retrieval (ACM SIGIR)*.

ChengXiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342.

Maximum Expected F-Measure Training of Logistic Regression Models

Martin Jansche
Center for Computational Learning Systems
Columbia University
New York, NY 10027, USA
jansche@acm.org

Abstract

We consider the problem of training logistic regression models for binary classification in information extraction and information retrieval tasks. Fitting probabilistic models for use with such tasks should take into account the demands of the task-specific utility function, in this case the well-known F-measure, which combines recall and precision into a global measure of utility. We develop a training procedure based on empirical risk minimization / utility maximization and evaluate it on a simple extraction task.

1 Introduction

Log-linear models have been used in many areas of Natural Language Processing (NLP) and Information Retrieval (IR). Scenarios in which log-linear models have been applied often involve simple binary classification decisions or probability assignments, as in the following three examples: Ratnaparkhi et al. (1994) consider a restricted form of the prepositional phrase attachment problem where attachment decisions are binary; Ittycheriah et al. (2003) reduce entity mention tracking to the problem of modeling the probability of two mentions being linked; and Greiff and Ponte (2000) develop models of probabilistic information retrieval that involve binary decisions of relevance. What is common to all three approaches is the application of log-linear models to binary classification tasks.[1] As Ratnaparkhi (1998, p. 27f.) points out, log-linear models of binary response variables are equivalent to, and in fact mere notational variants of, logistic regression models.

In this paper we focus on binary classification tasks, and in particular on the loss or utility associated with classification decisions. The three problems mentioned before – prepositional phrase attachment, entity mention linkage, and relevance of a document to a query – differ in one crucial aspect: The first is evaluated in terms of accuracy or, equivalently, symmetric zero–one loss; but the second and third are treated as information extraction/retrieval problems and evaluated in terms of recall and precision. Recall and precision are combined into a single overall utility function, the well-known F-measure. It may be desirable to estimate the parameters of a logistic regression model by maximizing F-measure during training. This is analogous, and in a certain sense equivalent, to empirical risk minimization, which has been used successfully in related areas, such as speech recognition (Rahim and Lee, 1997), language modeling (Paciorek and Rosenfeld, 2000), and machine translation (Och, 2003).

The novel contribution of this paper is a training procedure for (approximately) maximizing the expected F-measure of a probabilistic classifier based on a logistic regression model. We formulate a vector-valued utility function which has a well-defined expected value; F-measure is then a rational function of this expectation and can be maximized numerically under certain conventional regularizing assumptions.

[1] These kinds of log-linear models are also known among the NLP community as "maximum entropy models" (Berger et al., 1996; Ratnaparkhi, 1998). This is an unfortunate choice of terminology, because the term "maximum entropy" does not uniquely determine a family of models unless the constraints subject to which entropy is being maximized are specified.

We begin with a review of logistic regression (Section 2) and then discuss the use of F-measure for evaluation (Section 3). We reformulate F-measure as a function of an expected utility (Section 4) which is maximized during training (Section 5). We discuss the differences between our parameter estimation technique and maximum likelihood training on a toy example (Section 6) as well as on a real extraction task (Section 7). We conclude with a discussion of further applications and generalizations (Section 8).

2 Review of Logistic Regression

Bernoulli regression models are conditional probability models of a binary response variable Y given a vector $\vec{X}$ of k explanatory variables $(X_1,\ldots,X_k)$. We will use the convention[2] that Y takes on a value $y \in \{-1,+1\}$.

Logistic regression models (Cox, 1958) are perhaps best viewed as instances of generalized linear models (Nelder and Wedderburn, 1972; McCullagh and Nelder, 1989) where the the response variable follows a Bernoulli distribution and the link function is the logit function. Let us summarize this first, before expanding the relevant definitions:

$$\begin{aligned} Y &\sim \text{Bernoulli}(p) \\ \text{logit}(p) &= \theta_0 + x_1\,\theta_1 + x_2\,\theta_2 + \cdots + x_k\,\theta_k \end{aligned}$$

What this means is that there is an unobserved quantity p, the success probability of the Bernoulli distribution, which we interpret as the probability that Y will take on the value $+1$:

$$\Pr(Y = +1 \mid \vec{X} = (x_1, x_2, \ldots, x_k), \vec{\theta}) = p$$

The logit (log odds) function is defined as follows:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

The logit function is used to transform a probability, constrained to fall within the interval $(0,1)$, into a real number ranging over $(-\infty,+\infty)$. The inverse function of the logit is the cumulative distribution function of the standard logistic distribution (also known as the *sigmoid* or *logistic* function), which we call g:

$$g(z) = \frac{1}{1+\exp(-z)}$$

This allows us to write

$$p = g(\theta_0 + x_1\,\theta_1 + x_2\,\theta_2 + \cdots + x_k\,\theta_k).$$

We also adopt the usual convention that $\vec{x} = (1, x_1, x_2, \ldots, x_k)$, which is a $k+1$-dimensional vector whose first component is always 1 and whose remaining k components are the values of the k explanatory variables. So the Bernoulli probability can be expressed as

$$p = g\left(\sum_{j=0}^{k} x_j\,\theta_j\right) = g\left(\vec{x}\cdot\vec{\theta}\right).$$

The conditional probability model then takes the following abbreviated form, which will be used throughout the rest of this paper:

$$\Pr(+1 \mid \vec{x}, \vec{\theta}) = \frac{1}{1+\exp(-\vec{x}\cdot\vec{\theta})} \qquad (1)$$

A classifier can be constructed from this probability model using the MAP decision rule. This means predicting the label $+1$ if $\Pr(+1 \mid \vec{x}, \vec{\theta})$ exceeds $1/2$, which amounts to the following:

$$y_{\text{map}}(\vec{x}) = \underset{y}{\text{argmax}}\ \Pr(y \mid \vec{x}, \vec{\theta}) = \text{sgn}\left(\vec{x}\cdot\vec{\theta}\right)$$

This illustrates the well-known result that a MAP classifier derived from a logistic regression model is equivalent to a (single-layer) perceptron (Rosenblatt, 1958) or linear threshold unit.

3 F-Measure

Suppose the parameter vector θ of a logistic regression model is known. The performance of the resulting classifier can then be evaluated in terms of the *recall* (or *sensitivity*) and *precision* of the classifier on an evaluation dataset. Recall (R) and precision (P) are defined in terms of the number of true positives (A), misses (B), and false alarms (C) of the classifier (cf. Table 1):

$$R = \frac{A}{A+B} \qquad\qquad P = \frac{A}{A+C}$$

[2]The natural choice may seem to be for Y to range over the set $\{0,1\}$, but the convention adopted here is more common for classification problems and has certain advantages which will become clear soon.

		predicted		total
		$+1$	-1	
true	$+1$	A	B	n_{pos}
	-1	C	D	n_{neg}
total		m_{pos}	m_{neg}	n

Table 1: Schema for a 2×2 contingency table

The F_α measure – familiar from Information Retrieval – combines recall and precision into a single utility criterion by taking their α-weighted harmonic mean:

$$F_\alpha(R,P) = \left(\alpha \frac{1}{R} + (1-\alpha)\frac{1}{P}\right)^{-1}$$

The F_α measure can be expressed in terms of the triple (A,B,C) as follows:

$$F_\alpha(A,B,C) = \frac{A}{A + \alpha B + (1-\alpha) C} \tag{2}$$

In order to define A, B, and C formally, we use the notation $[\![\pi]\!]$ to denote a variant of the Kronecker delta defined like this, where π is a Boolean expression:

$$[\![\pi]\!] = \begin{cases} 1 & \text{if } \pi \\ 0 & \text{if } \neg\pi \end{cases}$$

Given an evaluation dataset $(\vec{x}_1,y_1),\ldots,(\vec{x}_n,y_n)$ the counts of hits (true positives), misses, and false alarms are, respectively:

$$A = \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\, [\![y_i = +1]\!]$$
$$B = \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = -1]\!]\, [\![y_i = +1]\!]$$
$$C = \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\, [\![y_i = -1]\!]$$

Note that F-measure is seemingly a global measure of utility that applies to an evaluation dataset as a whole: while the F-measure of a classifier evaluated on a single supervised instance is well defined, the overall F-measure on a larger dataset is not a function of the F-measure evaluated on each instance in the dataset. This is in contrast to ordinary loss/utility, whose grand total (or average) on a dataset can be computed by direct summation.

4 Relation to Expected Utility

We reformulate F-measure as a scalar-valued rational function composed with a vector-valued utility function. This allows us to define notions of expected and average utility, setting up the discussion of parameter estimation in terms of empirical risk minimization (or rather, utility maximization).

Define the following vector-valued utility function u, where $u(\tilde{y} \mid y)$ is the utility of choosing the label $\tilde{y}$ if the true label is y:

$$\begin{aligned} u(+1 \mid +1) &= (1,0,0) \\ u(-1 \mid +1) &= (0,1,0) \\ u(+1 \mid -1) &= (0,0,1) \\ u(-1 \mid -1) &= (0,0,0) \end{aligned}$$

This function indicates whether a classification decision is a hit, miss, or false alarm. Correct rejections are not counted.

Expected values are, of course, well-defined for vector-valued functions. For example, the expected utility is

$$\mathrm{E}[u] = \sum_{(\vec{x},y)} u(y_{\text{map}}(\vec{x}) \mid y)\ \Pr(\vec{x},y).$$

In empirical risk minimization we approximate the expected utility of a classifier by its average utility U_S on a given dataset $S = (\vec{x}_1,y_1),\ldots,(\vec{x}_n,y_n)$:

$$\begin{aligned} \mathrm{E}[u] \approx U_S &= \frac{1}{n}\sum_{i=1}^{n} u(y_{\text{map}}(\vec{x}_i) \mid y_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} u(+1 \mid y_i)\, [\![y_{\text{map}}(\vec{x}_i) = +1]\!] \\ &\qquad + u(-1 \mid y_i)\, [\![y_{\text{map}}(\vec{x}_i) = -1]\!] \end{aligned}$$

Now it is easy to see that U_S is the following vector:

$$U_S = \frac{1}{n}\begin{pmatrix} \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\, [\![y_i = +1]\!] \\ \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = -1]\!]\, [\![y_i = +1]\!] \\ \sum_{i=1}^{n} [\![y_{\text{map}}(\vec{x}_i) = +1]\!]\, [\![y_i = -1]\!] \end{pmatrix} \tag{3}$$

So $U_S = n^{-1}(A,B,C)$ where A, B, and C are as defined before. This means that we can interpret the

F-measure of a classifier as a simple rational function of its empirical average utility (the scaling factor $1/n$ in (3) can in fact be omitted). This allows us to approach the parameter estimation task as an empirical risk minimization or utility maximization problem.

5 Discriminative Parameter Estimation

In the preceding two sections we assumed that the parameter vector $\vec{\theta}$ was known. Now we turn to the problem of estimating $\vec{\theta}$ by maximizing the F-measure formulated in terms of expected utility. We make the dependence on $\vec{\theta}$ explicit in the formulation of the optimization task:

$$\vec{\theta}^\star = \operatorname*{argmax}_{\vec{\theta}} F_\alpha(A(\vec{\theta}), B(\vec{\theta}), C(\vec{\theta})),$$

where $(A(\vec{\theta}), B(\vec{\theta}), C(\vec{\theta})) = U_S(\vec{\theta})$ as defined in (3). We encounter the usual problem: the basic quantities involved are integers (counts of hits, misses, and false alarms), and the optimization objective is a piecewise-constant functions of the parameter vector $\vec{\theta}$, due to the fact that $\vec{\theta}$ occurs exclusively inside Kronecker deltas. For example:

$$[\![y_{\text{map}}(\vec{x}) = +1]\!] = \left[\!\!\left[\Pr(+1 \mid \vec{x}, \vec{\theta}) > 0.5 \right]\!\!\right]$$

In general, we can set

$$\left[\!\!\left[\Pr(+1 \mid \vec{x}, \vec{\theta}) > 0.5 \right]\!\!\right] \approx \Pr(+1 \mid \vec{x}, \vec{\theta}), \qquad (4)$$

and in the case of logistic regression this arises as a special case of approximating the limit

$$\left[\!\!\left[\Pr(+1 \mid \vec{x}, \vec{\theta}) > 0.5 \right]\!\!\right] = \lim_{\gamma \to \infty} g(\gamma\, \vec{x} \cdot \vec{\theta})$$

with a fixed value of $\gamma = 1$. The choice of γ does not matter much. The important point is that we are now dealing with approximate quantities which depend continuously on $\vec{\theta}$. In particular $A(\vec{\theta}) \approx \tilde{A}(\vec{\theta})$, where

$$\tilde{A}(\vec{\theta}) = \sum_{\substack{i=1 \\ y_i=+1}}^{n} g(\gamma\, \vec{x}_i \cdot \vec{\theta}). \qquad (5)$$

Since the marginal total of positive instances n_{pos} (cf. Table 1) does not depend on $\vec{\theta}$, we use the identities $\tilde{B}(\vec{\theta}) = n_{\text{pos}} - \tilde{A}(\vec{\theta})$ and $\tilde{m}_{\text{pos}}(\vec{\theta}) = \tilde{A}(\vec{\theta}) + \tilde{C}(\vec{\theta})$ to rewrite the optimization objective as $\tilde{F}_\alpha$:

$$\tilde{F}_\alpha(\vec{\theta}) = \frac{\tilde{A}(\vec{\theta})}{\alpha\, n_{\text{pos}} + (1-\alpha)\, \tilde{m}_{\text{pos}}(\vec{\theta})}, \qquad (6)$$

where $\tilde{A}(\vec{\theta})$ is given by (5) and $\tilde{m}_{\text{pos}}(\vec{\theta})$ is

$$\tilde{m}_{\text{pos}}(\vec{\theta}) = \sum_{i=1}^{n} g(\gamma\, \vec{x}_i \cdot \vec{\theta}).$$

Maximization of $\tilde{F}$ as defined in (6) can be carried out numerically using multidimensional optimization techniques like conjugate gradient search (Fletcher and Reeves, 1964) or quasi-Newton methods such as the BFGS algorithm (Broyden, 1967; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). This requires the evaluation of partial derivatives. The jth partial derivative of $\tilde{F}$ is as follows:

$$\begin{aligned}
\frac{\partial \tilde{F}(\vec{\theta})}{\partial \theta_j} &= h\, \frac{\partial \tilde{A}(\vec{\theta})}{\partial \theta_j} - h^2\, \tilde{A}(\vec{\theta})\, (1-\alpha)\, \frac{\partial \tilde{m}_{\text{pos}}(\vec{\theta})}{\partial \theta_j} \\
h &= \frac{1}{\alpha\, n_{\text{pos}} + (1-\alpha)\, \tilde{m}_{\text{pos}}(\vec{\theta})} \\
\frac{\partial \tilde{A}(\vec{\theta})}{\partial \theta_j} &= \sum_{\substack{i=1 \\ y_i=+1}}^{n} g'(\gamma\, \vec{x}_i \cdot \vec{\theta})\, \gamma x_{ij} \\
\frac{\partial \tilde{m}_{\text{pos}}(\vec{\theta})}{\partial \theta_j} &= \sum_{i=1}^{n} g'(\gamma\, \vec{x}_i \cdot \vec{\theta})\, \gamma x_{ij} \\
g'(z) &= g(z)\, (1 - g(z))
\end{aligned}$$

One can compute the value of $\tilde{F}(\vec{\theta})$ and its gradient $\nabla \tilde{F}(\vec{\theta})$ simultaneously at a given point $\vec{\theta}$ in $\mathrm{O}(nk)$ time and $\mathrm{O}(k)$ space. Pseudo-code for such an algorithm appears in Figure 1. In practice, the inner loops on lines 8–9 and 14–18 can be made more efficient by using a sparse representation of the row vectors $x[i]$. A concrete implementation of this algorithm can then be used as a callback to a multidimensional optimization routine. We use the BFGS minimizer provided by the GNU Scientific Library (Galassi et al., 2003). Important caveat: the function $\tilde{F}$ is generally not concave. We deal with this problem by taking the maximum across several runs of the optimization algorithm starting from random initial values. The next section illustrates this point further.

x	y
0	$+1$
1	-1
2	$+1$
3	$+1$

Table 2: Toy dataset

6 Comparison with Maximum Likelihood

A comparison with the method of maximum likelihood illustrates two important properties of discriminative parameter estimation. Consider the toy dataset in Table 2 consisting of four supervised instances with a single explanatory variable. Thus the logistic regression model has two parameters and takes the following form:

$$\Pr_{\text{toy}}(+1 \mid x, \theta_0, \theta_1) = \frac{1}{1 + \exp(-\theta_0 - x\,\theta_1)}$$

The log-likelihood function L is simply

$$\begin{aligned} L(\theta_0, \theta_1) = &\log \Pr_{\text{toy}}(+1 \mid 0, \theta_0, \theta_1) \\ &+ \log \Pr_{\text{toy}}(-1 \mid 1, \theta_0, \theta_1) \\ &+ \log \Pr_{\text{toy}}(+1 \mid 2, \theta_0, \theta_1) \\ &+ \log \Pr_{\text{toy}}(+1 \mid 3, \theta_0, \theta_1). \end{aligned}$$

A surface plot of L is shown in Figure 2. Observe that L is concave; its global maximum occurs near $(\theta_0, \theta_1) \approx (0.35, 0.57)$, and its value is always strictly negative because the toy dataset is not linearly separable. The classifier resulting from maximum likelihood training predicts the label $+1$ for all training instances and thus achieves a recall of $3/3$ and precision $3/4$ on its training data. The $F_{\alpha=0.5}$ measure is $6/7$.

Contrast the shape of the log-likelihood function L with the function $\tilde{F}_\alpha$. Surface plots of $\tilde{F}_{\alpha=0.5}$ and $\tilde{F}_{\alpha=0.25}$ appear in Figure 3. The figures clearly illustrate the first important (but undesirable) property of $\tilde{F}$, namely the lack of concavity. They also illustrate a desirable property, namely the ability to take into account certain properties of the loss function during training. The $\tilde{F}_{\alpha=0.5}$ surface in the left panel of Figure 3 achieves its maximum in the right corner for $(\theta_0, \theta_1) \rightarrow (+\infty, +\infty)$. If we choose $(\theta_0, \theta_1) = (20, 15)$ the classifier labels every instance of the training data with $+1$.

```
fdf($\theta$):
m ← 0
A ← 0
for j ← 0 to k do
    dm[j] ← 0
    dA[j] ← 0
for i ← 1 to n do
    p ← 0
    for j ← 0 to k do
        p ← p + x[i][j] × θ[j]
    p ← 1/(1 + exp(−d))
    m ← m + p
    if y[i] = +1 then
        A ← A + p
    for j ← 0 to k do
        t ← p × (1 − p) × x[i][j]
        dm[j] ← dm[j] + t
        if y[i] = +1 then
            dA[j] ← dA[j] + t
h ← 1/(α × n_pos + (1 − α) × m)
F ← h × A
t ← F × (1 − α)
for j ← 0 to k do
    dF[j] ← h × (dA[j] − t × dm[j])
return (F, dF)
```

Figure 1: Algorithm for computing $\tilde{F}$ and $\nabla\tilde{F}$

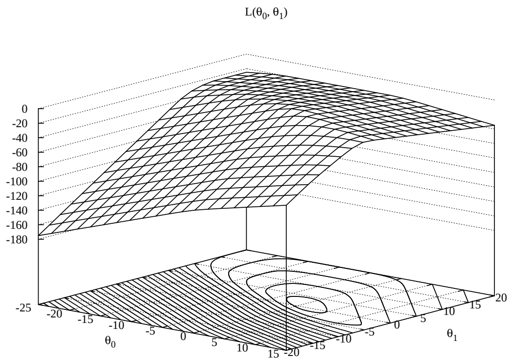
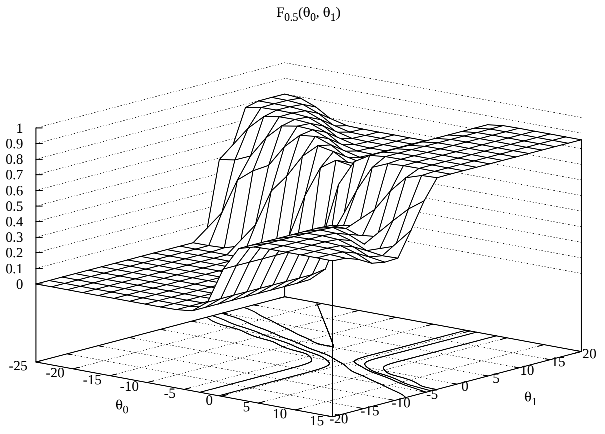


Figure 2: Surface plot of L on the toy dataset

Observe the difference between the $\tilde{F}_{\alpha=0.5}$ surface and the $\tilde{F}_{\alpha=0.25}$ surface in the right hand panel of Figure 3: $\tilde{F}_{\alpha=0.25}$ achieves its maximum in the back corner for $(\theta_0, \theta_1) \rightarrow (-\infty, +\infty)$. If we set $(\theta_0, \theta_1) = (-20, 15)$ the resulting classifier labels the first two



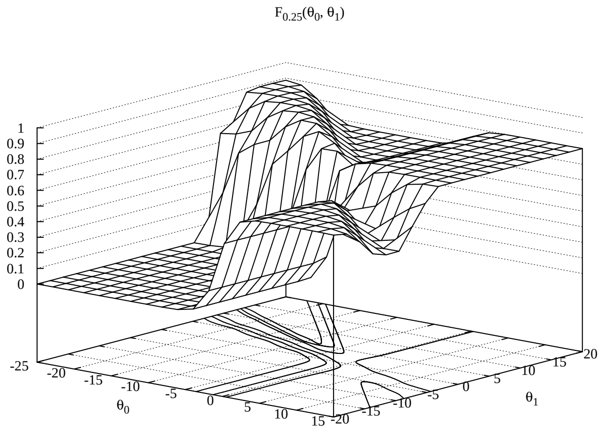


Figure 3: Surface plot of $\tilde{F}_{\alpha=0.5}$ (left) and $\tilde{F}_{\alpha=0.25}$ (right) on the toy dataset

instances ($x = 0$ and $x = 1$) as -1 and the last two instances ($x = 2$ and $x = 3$) as $+1$.

The classifier trained according to the $\tilde{F}_{\alpha=0.5}$ criterion achieves an $F_{\alpha=0.5}$ measure of $6/7 \approx 0.86$, compared with $4/5 = 0.80$ for the classifier trained according to the $\tilde{F}_{\alpha=0.25}$ criterion. Conversely, that classifier achieves an $F_{\alpha=0.25}$ measure of $8/9 \approx 0.89$ compared with $4/5 = 0.80$ for the classifier trained according to the $\tilde{F}_{\alpha=0.5}$ criterion. This demonstrates that the training procedure can effectively take information from the utility function into account, producing a classifier that performs well under a given evaluation criterion. This is the result of optimizing a task-specific utility function during training, not simply a matter of adjusting the decision threshold of a trained classifier.

7 Evaluation on an Extraction Problem

We evaluated our discriminative training procedure on a real extraction problem arising in broadcast news summarization. The overall task is to summarize the stories in an audio news broadcast (or in the audio portion of an A/V broadcast). We assume that story boundaries have been identified and that each story has been broken up into sentence-like units. A simple way of summarizing a story is then to classify each sentence as either belonging into a summary or not, so that a relevant subset of sentences can be extracted to form the basis of a summary. What makes the classification task hard, and therefore interesting, is the fact that reliable features are hard to come by. Existing approaches such as Maskey and Hirschberg 2005 do well only when combining diverse features such as lexical cues, acoustic properties, structural/positional features, etc.

The task has another property which renders it problematic, and which prompted us to develop the discriminative training procedure described in this paper. Summarization, by definition, aims for brevity. This means that in any dataset the number of positive instances will be much smaller than the number of negative instances. Given enough data, balance could be restored by discarding negative instances. This, however, was not an option in our case: a moderate amount of manually labeled data had been produced and about one third would have had to be discarded to achieve a balance in the distribution of class labels. This would have eliminated precious supervised training data, which we were not prepared to do.

The training and test data were prepared by Maskey and Hirschberg (2005), who performed the feature engineering, imputation of missing values, and the training–test split. We used the data unchanged in order to allow for a comparison between approaches. The dataset is made up of 30 features, divided into one binary response variable, and one binary explanatory variable plus 28 integer- and real-valued explanatory variables. The training portion consists of 3 535 instances, the test portion of 408 instances.

We fitted logistic regression models in three different ways: by maximum likelihood ML, by $\tilde{F}_{\alpha=0.5}$ maximization, and by $\tilde{F}_{\alpha=0.75}$ maximization. Each

Method	R	P	$F_{\alpha=0.5}$	$F_{\alpha=0.75}$
ML	24/99	24/33	0.3636	0.2909
ML†	85/99	85/229	0.5183	0.6464
$\tilde{F}_{\alpha=0.5}$	85/99	85/211	0.5484	0.6693
$\tilde{F}_{\alpha=0.75}$	95/99	95/330	0.4429	0.6061

Table 3: Evaluation results

classifier was evaluated on the test dataset and its recall (R), precision (P), $F_{\alpha=0.5}$ measure, and $F_{\alpha=0.75}$ measure recorded. The results appear in Table 3.

The row labeled ML† is special: the classifier used here is the logistic regression model fitted by maximum likelihood; what is different is that the threshold for positive predictions was adjusted *post hoc* to match the number of true positives of the first discriminatively trained classifier. This has the same effect as manually adjusting the threshold parameter θ_0 based on partial knowledge of the test data (via the performance of another classifier) and is thus not permissible. It is interesting to note, however, that the ML trained classifier performs worse than the $\tilde{F}_{\alpha=0.5}$ trained classifier even when one parameter is adjusted by an oracle with knowledge of the test data and the performance of the other classifier.

Fitting a model based on $\tilde{F}_{\alpha=0.75}$, which gives increased weight to recall compared with $\tilde{F}_{\alpha=0.5}$, led to higher recall as expected. However, we also expected that the $F_{\alpha=0.75}$ score of the $\tilde{F}_{\alpha=0.75}$ trained classifier would be higher than the $F_{\alpha=0.75}$ score of the $\tilde{F}_{\alpha=0.5}$ trained classifier. This is not the case, and could be due to the optimization getting stuck in a local maximum, or it may have been an unreasonable expectation to begin with.

8 Conclusions

We have presented a novel estimation procedure for probabilistic classifiers which we call, by a slight abuse of terminology, *maximum expected F-measure* training. We made use of the fact that expected utility computations can be carried out in a vector space, and that an ordering of vectors can be imposed for purposes of maximization which can employ auxiliary functions like the F-measure (2). This technique is quite general and well suited for working with other quantities that can be expressed in terms of hits, misses, false alarms, correct rejections, etc. In particular, it could be used to find a point estimate which provides a certain tradeoff between specificity and sensitivity, or operating point. A more general method would try to optimize several such operating points simultaneously, an issue which we will leave for future research.

The classifiers discussed in this paper are logistic regression models. However, this choice is not crucial. The approximation (4) is reasonable for binary decisions in general, and one can use it in conjunction with any well-behaved conditional Bernoulli model or related classifier. For Support Vector Machines, approximate F-measure maximization was introduced by Musicant et al. (2003).

Maximizing F-measure during training seems especially well suited for dealing with skewed classes. This can happen by accident, because of the nature of the problem as in our summarization example above, or by design: for example, one can expect skewed binary classes as the result of the one-vs-all reduction of multi-class classification to binary classification; and in multi-stage classification one may want to alternate between classifiers with high recall and classifiers with high precision.

Finally, the ability to incorporate non-standard tradeoffs between precision and recall at training time is useful in many information extraction and retrieval applications. Human end-users often create asymmetries between precision and recall, for good reasons: they may prefer to err on the side of caution (e.g., it is less of a problem to let an unwanted spam email reach a user than it is to hold back a legitimate message), or they may be better at some tasks than others (e.g., search engine users are good at filtering out irrelevant documents returned by a query, but are not equipped to crawl the web in order to look for relevant information that was not retrieved). In the absence of methods that work well for a wide range of operating points, we need training procedures that can be made sensitive to rare cases depending on the particular demands of the application.

Acknowledgements

I would like to thank Julia Hirschberg, Sameer Maskey, and the three anonymous reviewers for helpful comments. I am especially grateful to

Sameer Maskey for allowing me to use his speech summarization dataset for the evaluation in Section 7. The usual disclaimers apply.

References

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

C. G. Broyden. 1967. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381.

D. R. Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society, Series B (Methodological)*, 20(2):215–242.

R. Fletcher. 1970. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322. doi:10.1093/comjnl/13.3.317.

R. Fletcher and C. M. Reeves. 1964. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154. doi:10.1093/comjnl/7.2.149.

Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. 2003. *GNU Scientific Library Reference Manual*. Network Theory, Bristol, UK, second edition.

Donald Goldfarb. 1970. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26.

Warren R. Greiff and Jay M. Ponte. 2000. The maximum entropy approach and probabilistic IR models. *ACM Transactions on Information Systems*, 18(3):246–287. doi:10.1145/352595.352597.

Abraham Ittycheriah, Lucian Lita, Nanda Kambhatla, Nicolas Nicolov, Salim Roukos, and Margo Stys. 2003. Identifying and tracking entity mentions in a maximum entropy framework. In *HLT/NAACL 2003*. ACL Anthology N03-2014.

Sameer Maskey and Julia Hirschberg. 2005. Comparing lexical, acoustic/prosodic, structural and discourse features for speech summarization. In *Interspeech 2005 (Eurospeech)*.

P. McCullagh and J. A. Nelder. 1989. *Generalized Linear Models*. Chapman & Hall/CRC, Boca Raton, FL, second edition.

David R. Musicant, Vipin Kumar, and Aysel Ozgur. 2003. Optimizing F-measure with Support Vector Machines. In *FLAIRS 16*, pages 356–360.

J. A. Nelder and R. W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society, Series A (General)*, 135(3):370–384.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL 41*. ACL Anthology P03-1021.

Chris Paciorek and Roni Rosenfeld. 2000. Minimum classification error training in exponential language models. In *NIST/DARPA Speech Transcription Workshop*.

Mazin Rahim and Chin-Hui Lee. 1997. String-based minimum verification error (SB-MVE) training for speech recognition. *Computer, Speech and Language*, 11(2):147–160.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Computer and Information Science.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *ARPA Human Language Technology Workshop*, pages 250–255. ACL Anthology H94-1048.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

D. F. Shanno. 1970. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656.

Evita: A Robust Event Recognizer For QA Systems

Roser Saurí **Robert Knippen** **Marc Verhagen** **James Pustejovsky**
Lab for Linguistics and Computation
Computer Science Department
Brandeis University
415 South Street, Waltham, MA 02454, USA
{roser,knippen,marc,jamesp}@cs.brandeis.edu

Abstract

We present *Evita*, an application for recognizing events in natural language texts. Although developed as part of a suite of tools aimed at providing question answering systems with information about both temporal and intensional relations among events, it can be used independently as an event extraction tool. It is unique in that it is not limited to any pre-established list of relation types (events), nor is it restricted to a specific domain. Evita performs the identification and tagging of event expressions based on fairly simple strategies, informed by both linguistic- and statistically-based data. It achieves a performance ratio of 80.12% F-measure.[1]

1 Introduction

Event recognition is, after entity recognition, one of the major tasks within Information Extraction. It is currently being succesfully applied in different areas, like bioinformatics and text classification. Recognizing events in these fields is generally carried out by means of pre-defined sets of relations, possibly structured into an ontology, which makes such tasks domain dependent, but feasible. Event recognition is also at the core of Question Answering, since input questions touch on events and situations in the world (states, actions, properties, etc.), as they are reported in the text. In this field as well, the use of pre-defined sets of relation patterns has proved fairly reliable, particularly in the case of factoid type queries (Brill et al., 2002; Ravichandran and Hovy, 2002; Hovy et al., 2002; Soubbotin and Soubbotin, 2002).

Nonetheless, such an approach is not sensitive to certain contextual elements that may be fundamental for returning the appropriate answer. This is for instance the case in reporting or attempting contexts. Given the passage in (1a), a pattern-generated answer to question (1b) would be (1c). Similarly, disregarding the reporting context in example (2) could erroneously lead to concluding that no one from the White House was involved in the Watergate affair.

(1) a. Of the 14 known ways to reach the summit, only the East Ridge route has never been successfully climbed since George Mallory and Andrew ”Sandy” Irvine first attempted to climb Everest in 1924.
b. When did George Mallory and Andrew Irvine first climb Everest?
c. #In 1924.

(2) a. Nixon claimed that White House counsel John Dean had conducted an investigation into the Watergate matter and found that no-one from the White House was involved.
b. What members of the White House were involved in the Watergate matter?
c. #Nobody.

Intensional contexts like those above are generated by predicates referring to events of attempting, intending, commanding, and reporting, among others. When present in text, they function as modal

[1]This work was supported by a grant from the Advanced Research and Development Activity in Information Technology (ARDA), a U.S. Government entity which sponsors and promotes research of import to the Intelligence Community which includes but is not limited to the CIA, DIA, NSA, NIMA, and NRO.

qualifiers of the truth of a given proposition, as in example (2), or they indicate the factuality nature of the event expressed by the proposition (whether it happened or not), as in (1) (Saurí and Verhagen, 2005).

The need for a more sophisticated approach that sheds some awareness on the specificity of certain linguistic contexts is in line with the results obtained in previous TREC Question Answering competitions (Voorhees, 2002, 2003). There, a system that attempted a minimal understanding of both the question and the answer candidates, by translating them into their logical forms and using an inference engine, achieved a notably higher score than any surface-based system (Moldavan et al., 2002; Harabagiu et al., 2003).

Non-factoid questions introduce an even higher level of difficulty. Unlike factoid questions, there is no simple or unique answer, but more or less satisfactory ones instead. In many cases, they involve dealing with several events, or identifying and reasoning about certain relations among events which are only partially stated in the source documents (such as temporal and causal ones), all of which makes the pattern-based approach less suitable for the task (Small et al., 2003, Soricut and Brill, 2004). Temporal information in particular plays a significant role in the context of question answering systems (Pustejovsky et al., forthcoming). The question in (3), for instance, requires identifying a set of events related to the referred killing of peasants in Mexico, and subsequently ordering them along a temporal axis.

(3) What happened in Chiapas, Mexico, after the killing of 45 peasants in Acteal?

Reasoning about events in intensional contexts, or with event-ordering relations such as temporality and causality, is a requisite for any open-domain QA system aiming at both factoid and non-factoid questions. As a first step, this involves the identification of all relevant events reported in the source documents, so that later processing stages can locate intensional context boundaries and temporal relations among these events.

In this article, we present **Evita**, a tool for recognizing events in natural language texts. It has been developed as part of a suite of tools aimed at providing QA systems with information about both temporal and intensional relations between events; we anticipate, however, that it will be useful for other NLP tasks as well, such as narrative understanding, summarization, and the creation of factual databases from textual sources.

In the next section, we provide the linguistic foundations and technical details of our event recognizer tool. Section 3 gives the results and discusses them in the context of the task. We conclude in section 4, with an overview of Evita's main achievements and a brief discussion of future directions.

2 Evita, An Event Recognition Tool

Evita ('Events In Text Analyzer') is an event recognition system developed under the ARDA-funded TARSQI research framework. TARSQI is devoted to two complementary lines of work: (1) establishing a specification language, TimeML, aimed at capturing the richness of temporal and event related information in language (Pustejovsky et al., 2003a, forthcoming), and (2) the construction of a set of tools that perform tasks of identifying, tagging, and reasoning about eventive and temporal information in natural language texts (Pustejovsky and Gaizauskas, forthcoming, Mani, 2005; Mani and Schiffman, forthcoming; Verhagen, 2004; Verhagen et al., 2005; Verhagen and Knippen, forthcoming). Within TARSQI's framework, Evita's role is locating and tagging all event-referring expressions in the input text that can be temporally ordered.

Evita combines linguistic- and statistically-based techniques to better address all subtasks of event recognition. For example, the module devoted to recognizing temporal information that is expressed through the morphology of certain event expressions (such as tense and aspect) uses grammatical information (see section 2.4), whereas disambiguating nouns that can have both eventive and non-eventive interpretations is carried out by a statistical module (section 2.3).

The functionality of Evita breaks down into two parts: event identification and analysis of the event-based grammatical features that are relevant for temporal reasoning purposes. Both tasks rely on a preprocessing step which performs part-of-speech tagging and chunking, and on a module for clustering together chunks that refer to the same event. In the following subsection we provide the linguistic assumptions informing Evita. Then, subsections 2.2 to 2.5 provide a detailed description of Evita's different subcomponents: preprocessing, clustering of chunks, event identification, and analysis of the grammatical features associated to events.

2.1 Linguistic settings

TimeML identifies as events those event-denoting expressions that participate in the narrative of a given document and which can be temporally ordered. This includes all dynamic situations (punctual or durative) that happen or occur in the text, but also states in which something obtains or holds true, if they are temporally located in the text. As a result, generics and most state-denoting expressions are filtered out (see Saurí et al. (2004) for a more exhaustive definition of the criteria for event candidacy in TimeML).

Event-denoting expressions are found in a wide range of syntactic expressions, such as finite clauses (*that no-one from the White House* **was involved**), nonfinite clauses (*to* **climb** *Everest*), noun phrases headed by nominalizations (*the young industry's rapid* **growth**, *several anti-war* **demonstrations**) or event-referring nouns (*the controversial* **war**), and adjective phrases (*fully* **prepared**).

In addition to identifying the textual extent of events, Evita also analyzes certain grammatical features associated with them. These include:

- The *polarity* (positive or negative) of the expression tells whether the referred event has happened or not;
- *Modality* (as marked by modal auxiliaries *may, can, might, could, should*, etc., or adverbials like *probably, likely*, etc.) qualifies the denoted event with modal information (irrealis, necessity, possibility), and therefore has implications for the suitability of statements as answers to questions, in a parallel way to other intensional contexts exemplified in (1-2);
- *Tense* and *aspect* provide crucial information for the temporal ordering of the events;
- Similarly, the *non-finite morphology* of certain verbal expressions (infinitival, present participle, or past participle) has been shown as useful in predicting temporal relations between events (Lapata and Lascarides, 2004). We also consider as possible values here the categories of noun and adjective.
- Event *class* distinguishes among states (e.g., *be the director of*), general occurrences (*walk*), reporting (*tell*), intensional (*attempt*), and perception (*observe*) events. This classification is relevant for characterizing the nature of the event as irrealis, factual, possible, reported, etc. (recall examples (1-2) above).

Despite the fact that modality, tense, aspect, and non-finite morphology are typically verbal features, some nouns and adjectives can also have this sort of information associated with them; in particular, when they are part of the predicative complement of a copular verb (e.g., *may be* **ready**, *had been a* **collaborator**). A TimeML mark-up of these cases will tag only the complement as an event, disregarding the copular verb. Therefore, the modality, tense, aspect, and non-finite morphology information associated with the verb is incorporated as part of the event identified as the nominal or adjectival complement.

Except for *event class*, the characterization of all the features above relies strictly on surface linguistic cues. Notice that this surface-based approach does not provide for the actual temporal interpretation of the events in the given context. The tense of a verbal phrase, for example, does not always map in a straightforward way with the time being referred to in the world; e.g., simple present is sometimes used to express future time or habituality. We handle the task of mapping event features onto their semantics during a later processing stage, not addressed in this paper, but see Mani and Schiffman (forthcoming).

TimeML does not identify event participants, but the event tag and its attributes have been designed to interface with Named Entity taggers in a straightforward manner. In fact, the issue of argument linking to the events in TimeML is already being addressed in the effort to create a unified annotation with PropBank and NomBank (Pustejovsky et al. 2005). A complete overview of the linguistic foundations of TimeML can be obtained in Pustejovsky et al. (forthcoming).

2.2 Preprocessing

For the task of event recognition, Evita needs access to part of speech tags and to the result of some form of syntactic parsing. Section 2.1 above detailed some of the different syntactic structures that are used to refer to events. However, using a shallow parser is enough to retrieve event referring expressions, since they are generally conveyed by three possible part of speech categories: verbs (*go, see, say*), nouns (*departure, glimpse, war*), and adjectives (*upset, pregnant, dead*).

Part of speech tags and phrase chunks are also valuable for the identification of certain grammatical features such as tense, non-finite morphology, or polarity. Finally, lexical stems are necessary for those tasks involving lexical look-up. We obtain all such grammatical information by first preprocessing the input file using the Alembic Workbench tagger, lemmatizer, and chunker (Day et al., 1997). Evita's input must be XML-compliant, but need not conform to the TimeML DTD.

2.3 Event Recognition

Event identification in Evita is based on the notion of event as defined in the previous section. Only lexical items tagged by the preprocessing stage as either verbs, nouns, or adjectives are considered event candidates.

Different strategies are used for identifying events in these three categories. Event identification in verbal chunks is based on lexical look-up, accompanied by minimal contextual parsing in order to exclude weak stative predicates, such as 'be', and some generics (e.g., verbs with bare plural subjects). For every verbal chunk in the text, Evita first applies a pattern-based selection step that distinguishes among different kinds of information: the chunk head, which is generally the most-right element of verbal nature in the chunk, thus disregarding particles of different sort and punctuation marks; the modal auxiliary sequence, if any (e.g., *may have to*); the sequence of *do, have,* or *be* auxiliaries, marking for aspect, tense and voice; and finally, any item expressing the polarity of the event. The last three pieces of information will be used later, when identifying the event grammatical features (section 2.4).

Based on basic lexical inventories, the chunk may then be rejected if the head belongs to a certain class. For instance, copular verbs are generally disregarded for event tagging, although they enter into a a process of chunk clustering, together with their predicative complement (see section 2.5).

The identification of nominal and adjectival events is also initiated by the step of information selection. For each noun and adjective chunk, their head and polarity markers, if any, are distinguished.

Identifying events expressed by nouns involves two parts: a phase of lexical lookup, and a disambiguation process. The lexical lookup aims at an initial filtering of candidates to nominal events. First, Evita checks whether the head of the noun chunk is an event in WordNet. We identified about 25 subtrees from WordNet where all synsets denote nominal events. One of these, the largest, is the tree underneath the synset that contains the word *event*. Other subtrees were selected by analyzing events in SemCor and TimeBank1.2[2] and mapping them to WordNet synsets. One example is the synset with the noun *phenomenon*. In some cases, exceptions are defined. For example, a noun in a subset subsumed by the *phenomenon* synset is not an event if it is also subsumed by the synset with the noun *cloud* (in other words, many phenomena are events but clouds are not).

If the result of lexical lookup is inconclusive (that is, if a nominal occurs in WN as both and event and a non-event), then a disambiguation step is applied. This process is based on rules learned by a Bayesian classifier trained on SemCor.

Finally, identifying events from adjectives takes a conservative approach of tagging as events only those adjectives that were annotated as such in TimeBank1.2, whenever they appear as the head of a predicative complement. Thus, in addition to the use of corpus-based data, the subtask relies again on a minimal contextual parsing capable of identifying the complements of copular predicates.

[2]TimeBank1.2 is our gold standard corpus of around 200 news report documents from various sources, annotated with TimeML temporal and event information. A previous version, TimeBank1.1, can be downloaded from `http://www.timeml.org/`. For additional information see Pustejovsky et al. (2003b).

2.4 Identification of Grammatical Features

Identifying the grammatical features of events follows different procedures, depending on the part of speech of the event-denoting expression, and whether the feature is explicitely realized by the morphology of such expressions.

In event-denoting expressions that contain a verbal chunk, *tense*, *aspect*, and *non-finite morphology* values are directly derivable from the morphology of this constituent, which in English is quite straightforward. Thus, the identification of these features is done by first extracting the verbal constituents from the verbal chunk (disregarding adverbials, punctuation marks, etc.), and then applying a set of over 140 simple linguistic rules, which define different possible verbal phrases and map them to their corresponding tense, aspect, and non-finite morphology values. Figure 1 illustrates the rule for verbal phrases of future tense, progressive aspect, which bear the modal form *have to* (as in, e.g., *Participants* **will have to be working** *on the same topics*):

```
[form in futureForm],
[form=='have'],
[form=='to', pos=='TO'],
[form=='be'], [pos=='VBG'],
==>
[tense='FUTURE',
aspect='PROGRESSIVE',
nf_morph='NONE']
```

Figure 1: Grammatical Rule

For event-denoting expressions containing no verbal chunk, tense and aspect is established as null (`'NONE'` value), and non-finite morphology is `'noun'` or `'adjective'`, depending on the part-of-speech of their head.

Modality and *polarity* are the two remaining morphology-based features identified here. Evita extracts the values of these two attributes using basic pattern-matching techniques over the approapriate verbal, nominal, or adjectival chunk.

On the other hand, the identification of event *class* cannot rely on linguistic cues such as the morphology of the expression. Instead, it requires a combination of lexical resource-based look-up and word sense disambiguation. At present, this task has been attempted only in a very preliminary way, by tagging events with the class that was most frequently assigned to them in TimeBank1.2. Despite the limitations of such a treatment, the accuracy ratio is fairly good (refer to section 3).

2.5 Clustering of Chunks

In some cases, the chunker applied at the preprocessing stage identifies two independent constituents that contribute information about the same event. This may be due to a chunker error, but it is also systematically the case in verbal phrases containing the *have to* modal form or the *be going to* future form (Figure 2).

```
<VG>
<VX><lex pos="VBD">had</lex></VX>
</VG>
<VG-INF>
<INF><lex pos="TO">to</lex>
<lex pos="VB">say</lex>
</INF>
</VG-INF>
```

Figure 2: *have to* VP

It may be also necessary in verbal phrases with other modal auxiliaries, or with auxiliary forms of the *have*, *do*, or *be* forms, in which the auxiliary part is split off the main verb because of the presence of an adverbial phrase or similar (Figure 3).

```
<VG>
<VX><lex pos="VBZ">has</lex></VX>
</VG>
<lex pos=",">,</lex>
<lex pos="IN">of</lex>
<NG>
<HEAD><lex pos="NN">course</lex></HEAD>
</NG>
<lex pos=",">,</lex>
<VG>
<VX><lex pos="VBD">tried</lex></VX>
</VG>
```

Figure 3: *have V_en* VP

Constructions with copular verbs are another kind of context which requires clustering of chunks, in order to group together the verbal chunk corresponding to the copular predicate and the non-verbal chunk that functions as its predicative complement. In all these cases, additional syntactic parsing is needed for the tasks of event recognition and grammatical feature identification, in order to cluster together the two independent chunks.

The task of clustering chunks into bigger ones is activated by specific triggers (e.g., a chunk headed by an auxiliary form, or a chunk headed by the copular verb *be*) and carried out locally in the context of that trigger. For each trigger, there is a set of grammatical patterns describing the possible structures it can be a constituent of. The form *have*, for instance, may be followed by an infinitival phrase *to V*, constituting part of the modal form *have to* in the bigger verbal group *have to V*, as in Figure 2 above, or it may also be followed by a past participle-headed chunk, with which it forms a bigger verbal phrase *have V-en* expressing perfective aspect (Figure 3).

The grammatical patterns established for each trigger are written using the standard syntax of regular expressions, allowing for a greater expressiveness in the description of sequences of chunks (optionality of elements, inclusion of adverbial phrases and punctuation marks, variability in length, etc.). These patterns are then compiled into finite state automata that work with grammatical objects instead of string characters. Such an approach is based on well-established techniques using finite-state methods (see for instance Koskenniemi, 1992; Appelt et al. 1993; Karttunen et al., 1996; Grefenstette, 1996, among others).

Evita sequentially feeds each of the FSAs for the current trigger with the right-side part of the trigger context (up to the first sentence boundary), which is represented as a sequence of grammatical objects. If one of the FSAs accepts this sequence or a subpart of it, then the clustering operation is applied on the chunks within the accepted (sub)sequence.

3 Results

Evaluation of Evita has been carried out by comparing its performance against TimeBank1.2. The current performance of Evita is at 74.03% precision, 87.31% recall, for a resulting F-measure of 80.12% (with β=0.5). These results are comparable to the interannotation agreement scores for the task of tagging verbal and nominal events, by graduate linguistics students with only basic training (Table 1).[3] By basic training we understand that they had read the guidelines, had been given some additional advice, and subsequently annotated over 10 documents before annotating those used in the interannotation evaluation. They did not, however, have any meetings amongst themselves in order to discuss issues or to agree on a common strategy.

[3]These figures are also in terms of F-measure. See Hripcsak and Rothschild (2005) for the use of such metric in order to quantify interannotator reliability.

Category	F-measure
Nouns	64%
Verbs	80%

Table 1: Interannotation Agreement

On the other hand, the Accuracy ratio (i.e., the percentage of values Evita marked according to the gold standard) on the identification of event grammatical features is as shown:

Feature	Accuracy
polarity	98.26%
aspect	97.87%
modality	97.02%
tense	92.05%
nf_morph	89.95%
class	86.26%

Table 2: Accuracy of Grammatical Features

Accuracy for *polarity, aspect*, and *modality* is optimal: over 97% in all three cases. In fact, we were expecting a lower accuracy for polarity, since Evita relies only on the polarity elements present in the chunk containg the event, but does not take into account non-local forms of expressing polarity in English, such as negative polarity on the subject of a sentence (as in *Nobody saw him* or in *No victims were found*).

The slightly lower ratio for *tense* and *nf_morph* is in most of the cases due to problems from the POS tagger used in the preprocessing step, since tense and non-finite morphology values are mainly based on its result. Some common POS tagging mistakes deriving on tense and nf_morph errors are, for instance, identifying a present form as the base form of the verb, a simple past form as a past participle form, or vice versa. Errors in the nf_morph value are also due to the difficulty in distinguishing sometimes between present participle and noun (for *ing*-forms), or between past participle and adjective.

The lowest score is for event *class*, which nevertheless is in the 80s%. This is the only feature that cannot be obtained based on surface cues. Evita's treatment of this feature is still very basic, and we envision that it can be easily enhanced by exploring standard word sense disambiguation techniques.

4 Discussion and Conclusions

We have presented Evita, a tool for recognizing and tagging events in natural language text. To our knowledge, this is a unique tool within the community, in that it is not based on any pre-established list of event patterns, nor is it restricted to a specific domain. In addition, Evita identifies the grammatical information that is associated with the event-referring expression, such as tense, aspect, polarity, and modality. The characterization of these features is based on explicit linguistic cues. Unlike other work on event recognition, Evita does not attempt to identify event participants, but relies on the use of entity taggers for the linking of arguments to events.

Evita combines linguistic- and statistically-based knowledge to better address each particular subtask of the event recognition problem. Linguistic knowledge has been used for the parsing of very local and controlled contexts, such as verbal phrases, and the extraction of morphologically explicit information. On the other hand, statistical knowledge has contributed to the process of disambiguation of nominal events, following the current trend in the Word Sense Disambiguation field.

Our tool is grounded on simple and well-known technologies; namely, a standard preprocessing stage, finite state techniques, and Bayesian-based techniques for word sense disambiguation. In addition, it is conceived from a highly modular perspective. Thus, an effort has been put on separating linguistic knowledge from the processing thread. In this way we guarantee a low-cost maintainance of the system, and simplify the task of enriching the grammatical knowledge (which can be carried out even by naive programmers such as linguists) when additional data is obtained from corpus exploitation.

Evita is a component within a larger suite of tools. It is one of the steps within a processing sequence which aims at providing basic semantic information (such as temporal relations or intensional context boundaries) to applications like Question Answering or Narrative Understanding, for which text understanding is shown to be fundamental, in addition to shallow-based techniques. Nonetheless, Evita can also be used independently for purposes other than those above.

Additional tools within the TimeML research framework are (a) GUTime, a recognizer of temporal expressions which extends Tempex for TimeML (Mani, 2005), (b) a tool devoted to the temporal ordering and anchoring of events (Mani and Schiffman, forthcoming), and (c) Slinket, an application in charge of identifying subordination contexts that introduce intensional events like those exemplified in (1-2) (Verhagen et al., 2005). Together with these, Evita provides capabilities for a more adequate treatment of temporal and intensional information in textual sources, thereby contributing towards incorporating greater inferential capabilities to applications within QA and related fields, a requisite that has been shown necessary in the Introduction section.

Further work on Evita will be focused on two main areas: (1) improving the sense disambiguation of candidates to event nominals by experimenting with additional learning techniques, and (2) improving event classification. The accuracy ratio for this latter task is already fairly acceptable (86.26%), but it still needs to be enhanced in order to guarantee an optimal detection of subordinating intensional contexts (recall examples 1-2). Both lines of work will involve the exploration and use of word sense disambiguation techniques.

References

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel and Mabry Tyson 1993. 'FASTUS: A Finite-state Processor for Information Extraction from Real-world Text'. *Proceedings IJCAI-93*.

Brill, Eric, Susan Dumais and Michele Banko. 2002. 'An Analysis of the AskMSR Question Answering System'. *Proceedings of EMNLP 2002*.

Day, David,, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson and Marc Vilain. 1997. 'Mixed-Initiative Development of Language Processing Systems'. *Fifth Conference on Applied Natural Language Processing Systems*: 88–95.

Grefenstette, Gregory. 1996. 'Light Parsing as Finite-State Filtering'. *Workshop on Extended Finite State Models of Language, ECAI'96*.

Harabagiu, S., D. Moldovan, C. Clark, M. Bowden, J. Williams and J. Bensley. 2003. 'Answer Mining by Combining Extraction Techniques with Abductive Reasoning'. *Proceedings of the Text Retrieval Conference, TREC 2003*: 375-382.

Hovy, Eduard, Ulf Hermjakob and Deepak Ravichandran. 2002. A Question/Answer Typology with Surface Text Patterns. *Proceedings of the Second International Conference on Human Language Technology Research, HLT 2002*: 247-251.

Hripcsak, George and Adam S. Rothschild. 2005. 'Agreement, the F-measure, and reliability in information retrieval'. *Journal of the American Medical Informatics Association*, 12: 296-298.

Karttunen, L., J-P. Chanod, G. Grefenstette and A. Schiller. 1996. 'Regular Expressions for Language Engineering'. *Natural Language Engineering*, 2(4).

Koskenniemi, Kimmo, Pasi Tapanainen and Atro Voutilainen. 'Compiling and Using Finite-State Syntactic Rules'. *Proceedings of COLING-92*: 156-162.

Lapata, Maria and Alex Lascarides 2004. Inferring Sentence-Internal Temporal Relations. *Proceedings of HLT-NAACL 2004*.

Mani, Inderjeet. 2005. *Time Expression Tagger and Normalizer.* http://complingone.georgetown.edu/ linguist/GU_TIME_DOWNLOAD.HTML

Mani, Inderjeet and Barry Schiffman. Forthcoming. 'Temporally Anchoring and Ordering Events in News'. James Pustejovsky and Robert Gaizauskas (eds.) *Event Recognition in Natural Language*. John Benjamins.

Moldovan, D., S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu and O. Bolohan. 2002. 'LCC Tools for Question Answering'. *Proceedings of the Text REtrieval Conference, TREC 2002*.

Pustejovsky, J., J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. *IWCS-5 Fifth International Workshop on Computational Semantics.*

Pustejovsky, James and Rob Gaizauskas (editors) (forthcoming) *Reasoning about Time and Events.* John Benjamins Publishers.

Pustejovsky, J., P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro and M. Lazo. 2003b. The TIMEBANK Corpus. *Proceedings of Corpus Linguistics 2003*: 647-656.

Pustejovsky, J., B. Knippen, J. Littman, R. Saurí (forthcoming) Temporal and Event Information in Natural language Text. *Language Resources and Evaluation*.

Pustejovsky, James, Martha Palmer and Adam Meyers. 2005. Workshop on Frontiers in Corpus Annotation II. Pie in the Sky. ACL 2005.

Pustejovsky, J., R. Saurí, J. Castaño, D. R. Radev, R. Gaizauskas, A. Setzer, B. Sundheim and G. Katz. 2004. Representing Temporal and Event Knowledge for QA Systems. Mark T. Maybury (ed.) *New Directions in Question Answering.* MIT Press, Cambridge.

Ravichandran, Deepak and Eduard Hovy. 2002. 'Learning Surface Text Patterns for a Question Answering System'. *Proceedings of the ACL 2002.*

Saurí, Roser, Jessica Littman, Robert Knippen, Rob Gaizauskas, Andrea Setzer and James Pustejovsky. 2004. *TimeML Annotation Guidelines*. http://www.timeml.org.

Saurí, Roser and Marc Verhagen. 2005. Temporal Information in Intensional Contexts. Bunt, H., J. Geertzen and E. Thijse (eds.) *Proceedings of the Sixth International Workshop on Computational Semantics*. Tilburg, Tilburg University: 404-406.

Small, Sharon, Liu Ting, Nobuyuki Shimuzu and Tomek Strzalkowski. 2003. HITIQA, An interactive question answering system: A preliminary report. *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering.*

Soricut, Radu and Eric Brill. 2004. Automatic Question Answering: Beyond the Factoid. *HLT-NAACL 2004, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*: 57-64.

Soubbotin, Martin M. and Sergei M. Soubbotin. 2002. 'Use of Patterns for Detection of Answer Strings: A Systematic Approach'. *Proceedings of TREC-11.*

Verhagen, Marc. 2004. *Times Between the Lines*. Ph.D. thesis. Brandeis University. Waltham, MA, USA.

Verhagen, Marc and Robert Knippen. Forthcoming. TANGO: A Graphical Annotation Environment for Ordering Relations. James Pustejovsky and Robert Gaizauskas (eds.) *Time and Event Recognition in Natural Language*. John Benjamin Publications.

Verhagen, Marc, Inderjeet Mani, Roser Saurí, Robert Knippen, Jess Littman and James Pustejovsky. 2005. 'Automating Temporal Annotation with TARSQI'. Demo Session. *Proceedings of the ACL 2005*.

Voorhees, Ellen M. 2002. 'Overview of the TREC 2002 Question Answering Track'. *Proceedings of the Eleventh Text REtrieval Conference, TREC 2002.*

Voorhees, Ellen M. 2003. 'Overview of the TREC 2003 Question Answering Track'. *Proceedings of 2003 Text REtrieval Conference, TREC 2003*.

Using Sketches to Estimate Associations

Ping Li
Department of Statistics
Stanford University
Stanford, California 94305
pingli@stat.stanford.edu

Kenneth W. Church
Microsoft Research
One Microsoft Way
Redmond, Washington 98052
church@microsoft.com

Abstract

We should not have to look at the entire corpus (e.g., the Web) to know if two words are associated or not.[1] A powerful sampling technique called *Sketches* was originally introduced to remove duplicate Web pages. We generalize sketches to estimate contingency tables and associations, using a maximum likelihood estimator to find the most likely contingency table given the sample, the margins (document frequencies) and the size of the collection. Not unsurprisingly, computational work and statistical accuracy (variance or errors) depend on sampling rate, as will be shown both theoretically and empirically. Sampling methods become more and more important with larger and larger collections. At Web scale, sampling rates as low as 10^{-4} may suffice.

1 Introduction

Word associations (co-occurrences) have a wide range of applications including: Speech Recognition, Optical Character Recognition and Information Retrieval (IR) (Church and Hanks, 1991; Dunning, 1993; Manning and Schutze, 1999). It is easy to compute association scores for a small corpus, but more challenging to compute lots of scores for lots of data (e.g. the Web), with billions of web pages (D) and millions of word types (V). For a small corpus, one could compute pair-wise associations by multiplying the (0/1) term-by-document matrix with its transpose (Deerwester et al., 1999). But this is probably infeasible at Web scale.

[1]This work was conducted at Microsoft while the first author was an intern. The authors thank Chris Meek, David Heckerman, Robert Moore, Jonathan Goldstein, Trevor Hastie, David Siegmund, Art Own, Robert Tibshirani and Andrew Ng.

Approximations are often good enough. We should not have to look at every document to determine that two words are strongly associated. A number of sampling-based randomized algorithms have been implemented at Web scale (Broder, 1997; Charikar, 2002; Ravichandran et al., 2005).[2]

A conventional random sample is constructed by selecting D_s documents from a corpus of D documents. The (corpus) sampling rate is $\frac{D_s}{D}$. Of course, word distributions have long tails. There are a few high frequency words and many low frequency words. It would be convenient if the sampling rate could vary from word to word, unlike conventional sampling where the sampling rate is fixed across the vocabulary. In particular, in our experiments, we will impose a floor to make sure that the sample contains at least 20 documents for each term. (When working at Web scale, one might raise the floor somewhat to perhaps 10^4.)

Sampling is obviously helpful at the top of the frequency range, but not necessarily at the bottom (especially if frequencies fall below the floor). The question is: how about "ordinary" words? To answer this question, we randomly picked 15 pages from a Learners' dictionary (Hornby, 1989), and selected the first entry on each page. According to Google, there are 10 million pages/word (median value, aggregated over the 15 words), no where near the floor.

Sampling can make it possible to work in memory, avoiding disk. At Web scale ($D \approx 10$ billion pages), inverted indexes are large (1500 GBs/billion pages)[3], probably too large for memory. But a sample is more manageable; the inverted index for a 10^{-4} sample of the entire web could fit in memory on a single PC (1.5 GB).

[2]*http://labs.google.com/sets* produces fascinating sets, although we don't know how it works. Given the seeds, "America" and "China," *http://labs.google.com/sets* returns: "America, China, Japan, India, Italy, Spain, Brazil, Persia, Europe, Australia, France, Asia, Canada."

[3]This estimate is extrapolated from Brin and Page (1998), who report an inverted index of 37.2 GBs for 24 million pages.

Table 1: The number of intermediate results after the first join can be reduced from 504,000 to 120,000, by starting with "Schwarzenegger & Austria" rather than the baseline ("Schwarzenegger & Terminator"). The standard practice of starting with the two least frequent terms is a good rule of thumb, but one can do better, given (estimates of) joint frequencies.

Query	Hits (Google)
Austria	88,200,000
Governor	37,300,000
Schwarzenegger	4,030,000
Terminator	3,480,000
Governor & Schwarzenegger	1,220,000
Governor & Austria	708,000
Schwarzenegger & Terminator	504,000
Terminator & Austria	171,000
Governor & Terminator	132,000
Schwarzenegger & Austria	120,000

1.1 An Application: The Governator

Google returns the top k hits, plus an estimate of how many hits there are. Table 1 shows the number of hits for four words and their pair-wise combinations. Accurate estimates of associations would have applications in Database query planning (Garcia-Molina et al., 2002). Query optimizers construct a plan to minimize a cost function (e.g., intermediate writes). The optimizer could do better if it could estimate a table like Table 1. But efficiency is important. We certainly don't want to spend more time optimizing the plan than executing it.

Suppose the optimizer wanted to construct a plan for the query: "Governor Schwarzenegger Terminator Austria." The standard solution starts with the two least frequent terms: "Schwarzenegger" and "Terminator." That plan generates 504,000 intermediate writes after the first join. An improvement starts with "Schwarzenegger" with "Austria," reducing the 504,000 down to 120,000.

In addition to counting hits, Table 1 could also help find the top k pages. When joining the first pair of terms, we'd like to know how far down the ranking we should go. Accurate estimates of associations would help the optimizer make such decisions.

It is desirable that estimates be consistent, as well as accurate. Google, for example, reports 6 million hits for "America, China, Britain," and 23 million for "America, China, Britain, Japan." Joint frequencies decrease monotonically: $s \subset S \implies \text{hits}(s) \geq \text{hits}(S)$.

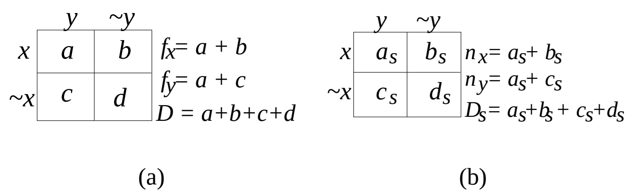


Figure 1: (a): A contingency table for word x and word y. Cell a is the number of documents that contain both x and y, b is the number that contain x but not y, c is the number that contain y but not x, and d is the number that contain neither x nor y. The margins, $f_x = a + b$ and $f_y = a + c$ are known as document frequencies in IR. D is the total number of documents in the collection. (b): A sample contingency table, with "s" indicating the *sample space*.

1.2 Sampling and Estimation

Two-way associations are often represented as two-way contingency tables (Figure 1(a)). Our task is to construct a sample contingency table (Figure 1(b)), and estimate 1(a) from 1(b). We will use a maximum likelihood estimator (MLE) to find the most likely contingency table, given the sample and various other constraints. We will propose a sampling procedure that bridges two popular choices: (A) sampling over documents and (B) sampling over postings. The estimation task is straightforward and well-understood for (A). As we consider more flexible sampling procedures such as (B), the estimation task becomes more challenging.

Flexible sampling procedures are desirable. Many studies focus on rare words (Dunning, 1993; Moore, 2004); butterflies are more interesting than moths. The sampling rate can be adjusted on a word-by-word basis with (B), but not with (A). The sampling rate determines the trade-off between computational work and statistical accuracy.

We assume a standard inverted index. For each word x, there are a set of postings, X. X contains a set of document IDs, one for each document containing x. The size of postings, $f_x = |X|$, corresponds to the margins of the contingency tables in Figure 1(a), also known as document frequencies in IR.

The postings lists are approximated by *sketches*, skX, first introduced by Broder (1997) for removing duplicate web pages. Assuming that document IDs are random (e.g., achieved by a random permutation), we can compute skX, a random sample of

X, by simply selecting the first few elements of X.

In Section 3, we will propose using sketches to construct sample contingency tables. With this novel construction, the contingency table (and summary statistics based on the table) can be estimated using conventional statistical methods such as MLE.

2 Broder's Sketch Algorithm

One could randomly sample two postings and intersect the samples to estimate associations. The sketch technique introduced by Broder (1997) is a significant improvement, as demonstrated in Figure 2.

Assume that each document in the corpus of size D is assigned a unique random ID between 1 and D. The postings for word x is a sorted list of f_x doc IDs. The sketch, skX, is the first (smallest) s_x doc IDs in X. Broder used $\text{MIN}_s(Z)$ to denote the s smallest elements in the set, Z. Thus, $skX = \text{MIN}_{s_x}(X)$. Similarly, Y denotes the postings for word y, and skY denotes its sketch, $\text{MIN}_{s_y}(Y)$. Broder assumed $s_x = s_y = s$.

Broder defined resemblance (R) and sample resemblance (R_s) to be:

$$R = \frac{a}{a+b+c}, \quad R_s = \frac{|\text{MIN}_s(skX \cup skY) \cap skX \cap skY|}{|\text{MIN}_s(skX \cup skY)|}.$$

Broder (1997) proved that R_s is an unbiased estimator of R. One could use R_s to estimate a but he didn't do that, and it is not recommended.[4]

Sketches were designed to improve the coverage of a, as illustrated by Monte Carlo simulation in Figure 2. The figure plots, $\text{E}\left(\frac{a_s}{a}\right)$, percentage of intersections, as a function of (postings) sampling rate, $\frac{s}{f}$, where $f_x = f_y = f$, $s_x = s_y = s$. The solid lines (sketches), $\text{E}\left(\frac{a_s}{a}\right) \approx \frac{s}{f}$, are above the dashed curve (random sampling), $\text{E}\left(\frac{a_s}{a}\right) = \frac{s^2}{f^2}$. The difference is particularly important at low sampling rates.

3 Generalizing Sketches: $R \rightarrow$ Tables

Sketches were first proposed for estimating resemblance (R). This section generalizes the method to construct sample contingency tables, from which we can estimate associations: R, LLR, cosine, etc.

[4]There are at least three problems with estimating a from R_s. First, the estimate is biased. Secondly, this estimate uses just s of the $2 \times s$ samples; larger samples $\rightarrow$ smaller errors. Thirdly, we would rather not impose the restriction: $s_x = s_y$.

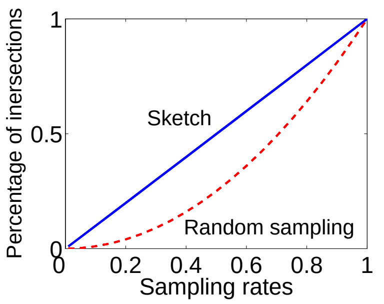


Figure 2: Sketches (solid curves) dominate random sampling (dashed curve). a=0.22, 0.38, 0.65, 0.80, 0.85f, f=0.2D, D=10^5. There is only one dashed curve across all values of a. There are different but indistinguishable solid curves depending on a.

Recall that the doc IDs span the integers from 1 to D with no gaps. When we compare two sketches, skX and skY, we have effectively looked at $D_s = \min\{skX_{(s_x)}, skY_{(s_y)}\}$ documents, where $skX_{(j)}$ is the jth smallest element in skX. The following construction generates the sample contingency table, a_s, b_s, c_s, d_s (as in Figure 1(b)). The example shown in Figure 3 may help explain the procedure.

$$\begin{aligned}
&D_s = \min\{skX_{(s_x)}, skY_{(s_y)}\}, \quad a_s = |skX \cap skY|,\\
&n_x = s_x - |\{j : skX_{(j)} > D_s\}|,\\
&n_y = s_y - |\{j : skY_{(j)} > D_s\}|,\\
&b_s = n_x - a_s, \quad c_s = n_y - a_s, \quad d_s = D_s - a_s - b_s - c_s.
\end{aligned}$$

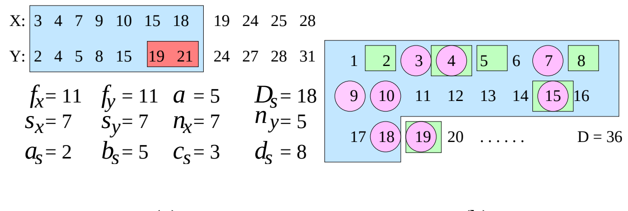
Given the sample contingency table, we are now ready to estimate the contingency table. It is sufficient to estimate a, since the rest of the table can be determined from f_x, f_y and D. For practical applications, we recommend the convenient closed-form approximation (8) in Section 5.1.

4 Margin-Free (MF) Baseline

Before considering the proposed MLE method, we introduce a baseline estimator that will not work as well because it does not take advantage of the margins. The baseline is the *multivariate hypergeometric* model, usually simplified as a *multinomial* by assuming "sample-with-replacement."

The sample expectations are (Siegrist, 1997),

$$\text{E}(a_s) = \frac{D_s}{D}a, \qquad \text{E}(b_s) = \frac{D_s}{D}b,$$
$$\text{E}(c_s) = \frac{D_s}{D}c, \qquad \text{E}(d_s) = \frac{D_s}{D}d. \tag{1}$$



X: 3 4 7 9 10 15 18 19 24 25 28
Y: 2 4 5 8 15 19 21 24 27 28 31
f_x = 11 f_y = 11 a = 5 D_s = 18
s_x = 7 s_y = 7 n_x = 7 n_y = 5
a_s = 2 b_s = 5 c_s = 3 d_s = 8
1 2 3 4 5 6 7 8
9 10 11 12 13 14 15 16
17 18 19 20 ...... D = 36
(a)
(b)

Figure 3: (a): The two sketches, skX and skY (larger shaded box), are used to construct a sample contingency table: a_s, b_s, c_s, d_s. skX consists of the first $s_x = 7$ doc IDs in X, the postings for word x. Similarly, skY consists of the first $s_y = 7$ doc IDs in Y, the postings for word y. There are 11 doc IDs in both X and Y, and $a = 5$ doc IDs in the intersection: {4, 15, 19, 24, 28}. (a) shows that $D_s = \min(18, 21) = 18$. Doc IDs 19 and 21 are excluded because we cannot determine if they are in the intersection or not, without looking outside the box. As it turns out, 19 is in the intersection and 21 is not. (b) enumerates the $D_s = 18$ documents, showing which documents contain x (small circles) and which contain y (small squares). Both procedures, (a) and (b), produce the same sample contingency table: $a_s = 2$, $b_s = 5$, $c_s = 3$ and $d_s = 8$.

The margin-free estimator and its variance are

$$\hat{a}_{MF} = \frac{D}{D_s}a_s, \;\; \text{Var}(\hat{a}_{MF}) = \frac{D}{D_s}\frac{1}{\frac{1}{a}+\frac{1}{D-a}}\frac{D-D_s}{D-1}. \quad (2)$$

For the multinomial simplification, we have

$$\hat{a}_{MF,r} = \frac{D}{D_s}a_s, \quad \text{Var}(\hat{a}_{MF,r}) = \frac{D}{D_s}\frac{1}{\frac{1}{a}+\frac{1}{D-a}}. \quad (3)$$

where "r" indicates "sample-with-replacement."

The term $\frac{D-D_s}{D-1} \approx \frac{D-D_s}{D}$ is often called the "finite-sample correction factor" (Siegrist, 1997).

5 The Proposed MLE Method

The task is to estimate the contingency table from the samples, the margins and D. We would like to use a maximum likelihood estimator for the most probable a, which maximizes the (full) likelihood (probability mass function, PMF) $P(a_s, b_s, c_s, d_s; a)$. Unfortunately, we do not know the exact expression for $P(a_s, b_s, c_s, d_s; a)$, but we do know the conditional probability $P(a_s, b_s, c_s, d_s|D_s; a)$. Since the doc IDs are uniformly random, sampling the first D_s contiguous documents is statistically equivalent to randomly sampling D_s documents from the corpus. Based on this key observation and Figure 3, *conditional* on D_s, $P(a_s, b_s, c_s, d_s|D_s; a)$ is the PMF of a two-way sample contingency table.

We factor the full likelihood into:

$$P(a_s, b_s, c_s, d_s; a) = P(a_s, b_s, c_s, d_s|D_s; a) \times P(D_s; a).$$

$P(D_s; a)$ is difficult. However, since we do not expect a strong dependency of D_s on a, we maximize the partial likelihood instead, and assume that is good enough. An example of partial likelihood is the Cox proportional hazards model in survival analysis (Venables and Ripley, 2002, Section 13.3) .

Our partial likelihood is

$$P(a_s, b_s, c_s, d_s|D_s; a) = \frac{\binom{a}{a_s}\binom{f_x-a}{b_s}\binom{f_y-a}{c_s}\binom{D-f_x-f_y+a}{d_s}}{\binom{D}{D_s}}$$
$$\propto \prod_{i=0}^{a_s-1}(a-i) \times \prod_{i=0}^{b_s-1}(f_x-a-i) \times \prod_{i=0}^{c_s-1}(f_y-a-i)$$
$$\times \prod_{i=0}^{d_s-1}(D-f_x-f_y+a-i), \quad (4)$$

where $\binom{n}{m} = \frac{n!}{m!(n-m)!}$. "$\propto$" is "proportional to."

We now derive an MLE for (4), a result that was not previously known, to the best of our knowledge. Let $\hat{a}_{MLE}$ maximizes $\log P(a_s, b_s, c_s, d_s|D_s; a)$:

$$\sum_{i=0}^{a_s-1}\log(a-i) + \sum_{i=0}^{b_s-1}\log(f_x-a-i)$$
$$+ \sum_{i=0}^{c_s-1}\log(f_y-a-i) + \sum_{i=0}^{d_s-1}\log(D-f_x-f_y+a-i),$$

whose first derivative, $\frac{\partial \log P(a_s,b_s,c_s,d_s|D_s;a)}{\partial a}$, is

$$\sum_{i=0}^{a_s-1}\frac{1}{a-i} - \sum_{i=0}^{b_s-1}\frac{1}{f_x-a-i} - \sum_{i=0}^{c_s-1}\frac{1}{f_y-a-i}$$
$$+ \sum_{i=0}^{d_s-1}\frac{1}{D-f_x-f_y+a-i}. \quad (5)$$

Since the second derivative, $\frac{\partial^2 \log P(a_s,b_s,c_s,d_s|D_s;a)}{\partial a^2}$, is negative, the log likelihood function is concave, hence has a unique maximum. One could numerically solve (5) for $\frac{\partial \log P(a_s,b_s,c_s,d_s|D_s;a)}{\partial a} = 0$. However, we derive the exact solution using the following updating formula from (4):

$$P(a_s,b_s,c_s,d_s|D_s;a) = P(a_s,b_s,c_s,d_s|D_s;a-1)\times \frac{f_x-a+1-b_s}{f_x-a+1}\frac{f_y-a+1-c_s}{f_y-a+1}\frac{D-f_x-f_y+a}{D-f_x-f_y+a-d_s} \frac{a}{a-a_s} = P(a_s,b_s,c_s,d_s|D_s;a-1)\times g(a). \tag{6}$$

Since our MLE is unique, it suffices to find a from $g(a)=1$, which is a cubic function in a.

5.1 A Convenient Practical Approximation

Rather than solving the cubic equation for the exact MLE, the following approximation may be more convenient. Assume we sample $n_x = a_s + b_s$ from X and obtain a_s co-occurrences without knowledge of the samples from Y. Further assuming "sample-with-replacement," a_s is then binomially distributed, $a_s \sim \text{Binom}(n_x, \frac{a}{f_x})$. Similarly, assume $a_s \sim \text{Binom}(n_y, \frac{a}{f_y})$. Under these assumptions, the PMF of a_s is a product of two binomial PMFs:

$$\binom{f_x}{n_x}\left(\frac{a}{f_x}\right)^{a_s}\left(\frac{f_x-a}{f_x}\right)^{b_s}\binom{f_y}{n_y}\left(\frac{a}{f_y}\right)^{a_s}\left(\frac{f_y-a}{f_y}\right)^{c_s} \propto a^{2a_s}(f_x-a)^{b_s}(f_y-a)^{c_s}. \tag{7}$$

Setting the first derivative of the logarithm of (7) to be zero, we obtain $\frac{2a_s}{a} - \frac{b_s}{f_x-a} - \frac{c_s}{f_y-a} = 0$, which is quadratic in a and has a solution:

$$\hat{a}_{MLE,a} = \frac{f_x(2a_s+c_s)+f_y(2a_s+b_s)}{2(2a_s+b_s+c_s)} - \frac{\sqrt{(f_x(2a_s+c_s)-f_y(2a_s+b_s))^2+4f_xf_yb_sc_s}}{2(2a_s+b_s+c_s)}. \tag{8}$$

Section 6 shows that $\hat{a}_{MLE,a}$ is very close to $\hat{a}_{MLE}$.

5.2 Theoretical Evaluation: Bias and Variance

How good are the estimates? A popular metric is mean square error (MSE): $\text{MSE}(\hat{a}) = \text{E}(\hat{a}-a)^2 = \text{Var}(\hat{a}) + \text{Bias}^2(\hat{a})$. If $\hat{a}$ is unbiased, $\text{MSE}(\hat{a}) = \text{Var}(\hat{a}) = \text{SE}^2(\hat{a})$, where SE is the standard error. Here all expectations are conditional on D_s.

Large sample theory (Lehmann and Casella, 1998, Chapter 6) says that, under "sample-with-replacement," $\hat{a}_{MLE}$ is asymptotically unbiased and converges to Normal with mean a and variance $\frac{1}{\text{I}(a)}$, where $\text{I}(a)$, the Fisher Information, is

$$\text{I}(a) = -\text{E}\left(\frac{\partial^2}{\partial a^2}\log P(a_s,b_s,c_s,d_s|D_s;a,r)\right). \tag{9}$$

Under "sample-with-replacement," we have

$$P(a_s,b_s,c_s,d_s|D_s;a,r) \propto \left(\frac{a}{D}\right)^{a_s}\times\left(\frac{f_x-a}{D}\right)^{b_s} \times\left(\frac{f_y-a}{D}\right)^{c_s}\times\left(\frac{D-f_x-f_y+a}{D}\right)^{d_s}, \tag{10}$$

Therefore, the Fisher Information, $\text{I}(a)$, is

$$\frac{\text{E}(a_s)}{a^2}+\frac{\text{E}(b_s)}{(f_x-a)^2}+\frac{\text{E}(c_s)}{(f_y-a)^2}+\frac{\text{E}(d_s)}{(D-f_x-f_y+a)^2}. \tag{11}$$

We plug (1) from the margin-free model into (11) as an approximation, to obtain

$$\text{Var}(\hat{a}_{MLE}) \approx \frac{\frac{D}{D_s}-1}{\frac{1}{a}+\frac{1}{f_x-a}+\frac{1}{f_y-a}+\frac{1}{D-f_x-f_y+a}}, \tag{12}$$

which is $\frac{1}{\text{I}(a)}$ multiplied by $\frac{D-D_s}{D}$, the "finite-sample correction factor," to consider "sample-without-replacement."

We can see that $\text{Var}(\hat{a}_{MLE})$ is less than $\text{Var}(\hat{a}_{MF})$ in (2). In addition, $\hat{a}_{MLE}$ is asymptotically unbiased while $\hat{a}_{MF}$ is no longer unbiased under margin constraints. Therefore, we expect $\hat{a}_{MLE}$ has smaller MSE than $\hat{a}_{MF}$. In other words, the proposed MLE method is more accurate than the MF baseline, in terms of variance, bias and mean square error. If we know the margins, we ought to use them.

5.3 Unconditional Bias and Variance

$\hat{a}_{MLE}$ is also unconditionally unbiased:

$$\text{E}(\hat{a}_{MLE}-a) = \text{E}(\text{E}(\hat{a}_{MLE}-a|D_s)) \approx \text{E}(0) = 0. \tag{13}$$

The unconditional variance is useful because often we would like to estimate the errors before knowing D_s (e.g., for choosing sample sizes).

To compute the unconditional variance of $\hat{a}_{MLE}$, we should replace $\frac{D}{D_s}$ with $\text{E}\left(\frac{D}{D_s}\right)$ in (12). We resort to an approximation for $\text{E}\left(\frac{D}{D_s}\right)$. Note that $skX_{(s_x)}$ is the order statistics of a discrete random variable (Siegrist, 1997) with expectation

$$\text{E}\left(skX_{(s_x)}\right) = \frac{s_x(D+1)}{f_x+1} \approx \frac{s_x}{f_x}D. \tag{14}$$

By Jensen's inequality, we know that

$$\text{E}\left(\frac{D_s}{D}\right) \le \min\left(\frac{\text{E}\left(skX_{(s_x)}\right)}{D}, \frac{\text{E}\left(skY_{(s_y)}\right)}{D}\right) = \min\left(\frac{s_x}{f_x}, \frac{s_y}{f_y}\right) \tag{15}$$

$$\text{E}\left(\frac{D}{D_s}\right) \ge \frac{1}{\text{E}\left(\frac{D_s}{D}\right)} \ge \max\left(\frac{f_x}{s_x}, \frac{f_y}{s_y}\right). \tag{16}$$

Table 2: Gold standard joint frequencies, a. Document frequencies are shown in parentheses. These words are frequent, suitable for evaluating our algorithms at very low sampling rates.

	THIS	HAVE	HELP	PROGRAM
THIS (27633)	—	13517	7221	3682
HAVE (17396)	13517	—	5781	3029
HELP (10791)	7221	5781	—	1949
PROGRAM (5327)	3682	3029	1949	—

Replacing the inequalities with equalities underestimates the variance, but only slightly.

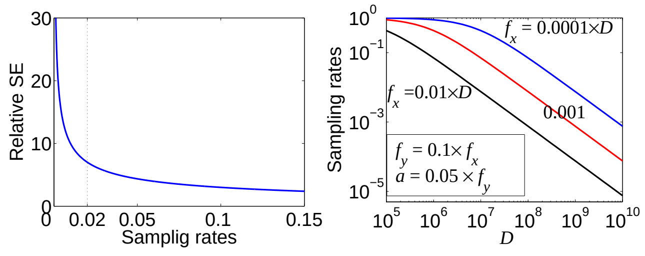
5.4 Smoothing

Although not a major emphasis here, our evaluations will show that $\hat{a}_{MLE+S}$, a smoothed version of the proposed MLE method, is effective, especially at low sampling rates. $\hat{a}_{MLE+S}$ uses "add-one" smoothing. Given that such a simple method is as effective as it is, it would be worth considering more sophisticated methods such as Good-Turing.

5.5 How Many Samples Are Sufficient?

The answer depends on the trade-off between computation and estimation errors. One simple rule is to sample "2%." (12) implies that the standard error is proportional to $\sqrt{D/D_s - 1}$. Figure 4(a) plots $\sqrt{D/D_s - 1}$ as a function of sampling rate, D_s/D, indicating a "elbow" about 2%. However, 2% is too large for high frequency words.

A more reasonable metric is the "coefficient of variation," cv = $\frac{\text{SE}(\hat{a})}{a}$. At Web scale (10 billion pages), we expect that a very small sampling rate such as 10^{-4} or 10^{-5} will suffice to achieve a reasonable cv (e.g., 0.5). See Figure 4(b).



(a) (b)

Figure 4: How large should the sampling rate be? (a): We can sample up to the "elbow point" (2%), but after that there are diminishing returns. (b): An analysis based on cv = $\frac{\text{SE}}{a}$ = 0.5 suggests that we can get away with much lower sampling rates. The three curves plot the critical value for the sampling rate, $\frac{D_s}{D}$, as a function of corpus size, D. At Web scale, $D \approx 10^{10}$, sampling rates above 10^{-3} to 10^{-5} satisfy cv < 0.5, at least for these settings of f_x, f_y and a. The settings were chosen to simulate "ordinary" words. The three curves correspond to three choices of f_x: $D/100$, $D/1000$, and $D/10,000$. $f_y = f_x/10$, $a = f_y/20$. SE is based on (12).

6 Evaluation

Two sets of experiments were run on a collection of $D = 2^{16}$ web pages, provided by MSN. The first experiment considered 4 English words shown in Table 2, and the second experiment considers 968 English words with mean *df* = 2135 and median *df* = 1135. They form 468,028 word pairs, with mean co-occurrences = 188 and median = 74.

6.1 Small Dataset Monte Carlo Experiment

Figure 5 evaluates the various estimate methods by MSE over a wide range of sampling rates. Doc IDs were randomly permuted 10^5 times. For each permutation we constructed sketches from the inverted index at a series of sampling rates. The figure shows that the proposed method, $\hat{a}_{MLE}$, is considerably better (by $20\% - 40\%$) than the margin-free baseline, $\hat{a}_{MF}$. Smoothing is effective at low sampling rates. The recommended approximation, $\hat{a}_{MLE,a}$, is remarkably close to the exact solution.

Figure 6 shows agreement between the theoretical and empirical unconditional variances. Smoothing reduces variances, at low sampling rates. We used the empirical $\text{E}\left(\frac{D}{D_S}\right)$ to compute the theoretical variances. The approximation, $\max\left(\frac{f_x}{s_x}, \frac{f_y}{s_y}\right)$, is $> 0.95\text{E}\left(\frac{D}{D_S}\right)$ at sampling rates > 0.01.

Figure 7 verifies that the proposed MLE is unbiased, unlike the margin-free baselines.

6.2 Large Dataset Experiment

The large experiment considers 968 English words (468,028 pairs) over a range of sampling rates. A floor of 20 was imposed on sample sizes.

As reported in Figure 8, the large experiment confirms once again that proposed method, $\hat{a}_{MLE}$, is considerably better than the margin-free baseline (by

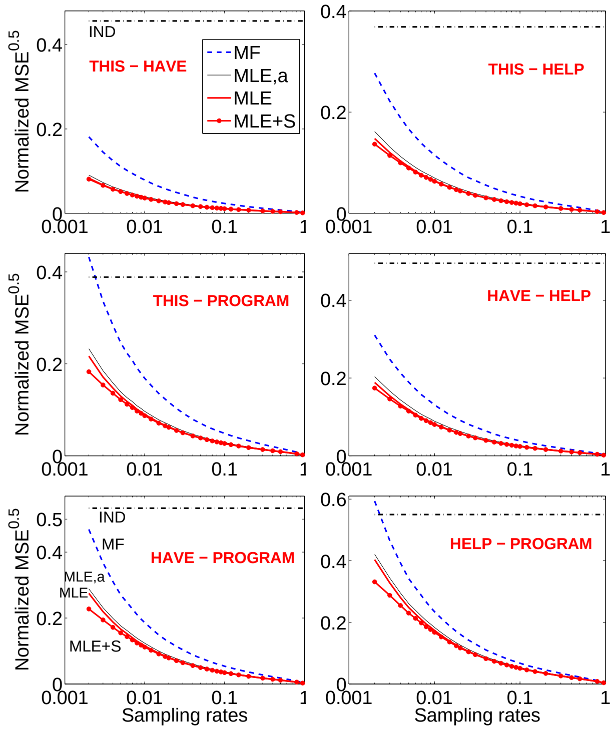


Figure 5: The proposed method, $\hat{a}_{MLE}$ outperforms the margin-free baseline, $\hat{a}_{MF}$, in terms of $\frac{\text{MSE}^{0.5}}{a}$. The recommended approximation, $\hat{a}_{MLE,a}$, is close to $\hat{a}_{MLE}$. Smoothing, $\hat{a}_{MLE+S}$, is effective at low sampling rates. All methods are better than assuming independence (IND).

$15\% - 30\%$). The recommended approximation, $\hat{a}_{MLE,a}$, is close to $\hat{a}_{MLE}$. Smoothing, $\hat{a}_{MLE+S}$ helps at low sampling rates.

6.3 Rank Retrieval: Top k Associated Pairs

We computed a gold standard similarity cosine ranking of the 468,028 pairs using a 100% sample: $\cos = \frac{a}{\sqrt{f_x f_y}}$. We then compared the gold standard to rankings based on smaller samples. Figure 9(a) compares the two lists in terms of agreement in the top k. For $3 \leq k \leq 200$, with a sampling rate of 0.005, the agreement is consistently 70% or higher. Increasing sampling rate, increases agreement.

The same comparisons are evaluated in terms of precision and recall in Figure 9(b), by fixing the top 1% of the gold standard list but varying the top percentages of the sample list. Again, increasing sampling rate, increases agreement.

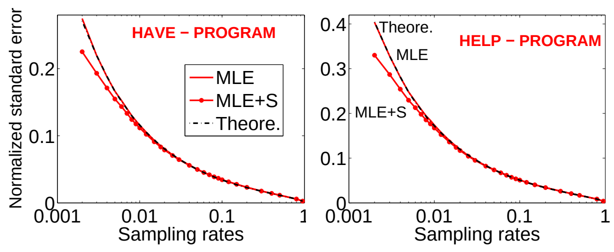


Figure 6: The theoretical and empirical variances show remarkable agreement, in terms of $\frac{\text{SE}(\hat{a})}{a}$. Smoothing reduces variances at low sampling rates.

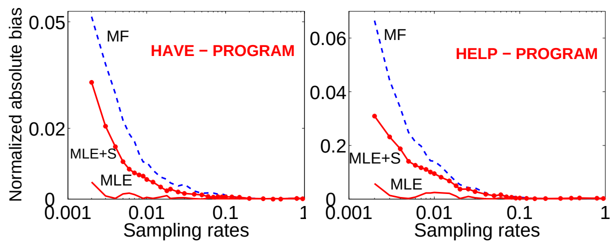


Figure 7: Biases in terms of $\frac{|\text{E}(\hat{a})-a|}{a}$. $\hat{a}_{MLE}$ is practically unbiased, unlike $\hat{a}_{MF}$. Smoothing increases bias slightly.

7 Conclusion

We proposed a novel sketch-based procedure for constructing sample contingency tables. The method bridges two popular choices: (A) sampling over documents and (B) sampling over postings. Well-understood maximum likelihood estimation (MLE) techniques can be applied to sketches (or to traditional samples) to estimate word associations. We derived an exact cubic solution, $\hat{a}_{MLE}$, as well as a quadratic approximation, $\hat{a}_{MLE,a}$. The approximation is recommended because it is close to the exact solution, and easy to compute.

The proposed MLE methods were compared empirically and theoretically to a margin-free (MF) baseline, finding large improvements. When we know the margins, we ought to use them.

Sample-based methods (MLE & MF) are often better than sample-free methods. Associations are often estimated without samples. It is popular to assume independence: (Garcia-Molina et al., 2002, Chapter 16.4), i.e., $\hat{a} \approx \frac{f_x f_y}{D}$. Independence led to large errors in our experiments.

Not unsurprisingly, there is a trade-off between computational work (space and time) and statistical

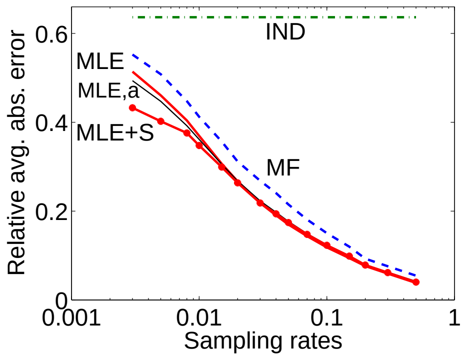


Figure 8: We report the (normalized) mean absolute errors (divided by the mean co-occurrences, 188). All curves are averaged over three permutations. The proposed MLE and the recommended approximation are very close and both are significantly better than the margin-free (MF) baseline. Smoothing, $\hat{a}_{MLE+S}$, helps at low sampling rates. All estimators do better than assuming independence.

accuracy (variance or errors); reducing the sampling rate saves work, but costs accuracy. We derived formulas for variance, showing precisely how accuracy depends on sampling rate. Sampling methods become more and more important with larger and larger collections. At Web scale, sampling rates as low as 10^{-4} may suffice for "ordinary" words.

We have recently generalized the sampling algorithm and estimation method to multi-way associations; see (Li and Church, 2005).

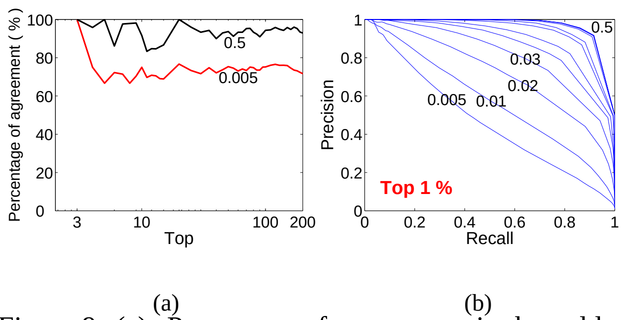


Figure 9: (a): Percentage of agreements in the gold standard and reconstructed (from samples) top 3 to 200 list. (b):Precision-recall curves in retrieving the top 1% gold standard pairs, at different sampling rates. For example, 60% recall and 70% precision is achieved at sampling rate = 0.02.

References

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, pages 107–117, Brisbane, Australia.

A. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29, Positano, Italy.

M. S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, Montreal, Quebec, Canada.

K. Church and P. Hanks. 1991. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29.

S. Deerwester, S. T. Dumais, G. W. Furnas, and T. K. Landauer. 1999. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

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

H. Garcia-Molina, J. D. Ullman, and J. D. Widom. 2002. *Database Systems: the Complete Book*. Prentice Hall, New York, NY.

A. S. Hornby, editor. 1989. *Oxford Advanced Learner's Dictionary*. Oxford University Press, Oxford, UK.

E. L. Lehmann and G. Casella. 1998. *Theory of Point Estimation*. Springer, New York, NY, second edition.

P. Li and K. W. Church. 2005. Using sketches to estimate two-way and multi-way associations. Technical report, Microsoft Research, Redmond, WA.

C. D. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

R. C. Moore. 2004. On log-likelihood-ratios and the significance of rare events. In *Proceedings of EMNLP 2004*, pages 333–340, Barcelona, Spain.

D. Ravichandran, P. Pantel, and E. Hovy. 2005. Randomized algorithms and NLP: Using locality sensitive hash function for high speed noun clustering. In *Proceedings of ACL*, pages 622–629, Ann Arbor.

K. Siegrist. 1997. *Finite Sampling Models*, *http://www.ds.unifi.it/VL/VL_EN/urn/index.html*. Virtual Laboratories in Probability and Statistics.

W. N. Venables and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Springer-Verlag, New York, NY, fourth edition.

Context and Learning in Novelty Detection

Barry Schiffman and **Kathleen R. McKeown**
Department of Computer Science
Columbia University
New York, N.Y.
{bschiff,kathy}@cs.columbia.edu

Abstract

We demonstrate the value of using context in a new-information detection system that achieved the highest precision scores at the Text Retrieval Conference's Novelty Track in 2004. In order to determine whether information within a sentence has been seen in material read previously, our system integrates information about the context of the sentence with novel words and named entities within the sentence, and uses a specialized learning algorithm to tune the system parameters.

1 Introduction

New-information detection addresses two important problems in a society awash in more digital information than people can exploit. A novelty detection system could help people who are tracking an event in the news, where numerous sources present similar material. It could also provide a way to organize summaries by focusing on the most recent information, much like an automated bulletin service.

We envision that many types of users would find such a system valuable. Certainly analysts, business people, and anyone interested in current events, would benefit from being able to track news stories automatically, without repetition. Different news organizations report on the same event, often working hard to make their reports look different from one another, whether or not they have new material to report. Our system would help readers to zero in on new information. In addition, a focus on new information provides a way of organizing a general summary.

Our approach is unique in representing and maintaining the focus in discourse. The idea stems from the fact that novelty often comes in bursts, which is not surprising since the articles are composed of some number of smaller, coherent segments. Each segment is started by some kind of introductory passage, and that is where we expect to find the *novel* words. Novel words are identified by comparing the current sentence's words against a table of all words seen in the inputs to that point. They let us know whether the entire segment is likely to contain more novel material. Subsequent passages are likely to continue the novel discussion whether or not they contain novel words. They may contain pronomial references or other anaphoric references to the novel entity. Our long-term goal is to integrate the approach described in this paper into our larger new-information detector, a system that performs a more complicated syntactic analysis of the input texts and employs machine learning to classify passages as new or old.

Meanwhile, we tested our focus-based approach at the Novelty Track at the Text Retrieval Conference (TREC) in 2004. The Novelty Tracks in 2003 and in 2004 were divided into four tasks; Task 1 and Task 3 incorporate retrieval, requiring submissions to locate the relevant sentences before filtering them for novelty. Tasks 2 and 4 are novelty detection alone, using the relevant sentences selected by humans as input. Since our interest is in novelty detection, we chose to concentrate on Task 2[1] Our TREC submission was also designed to test a specialized learning mechanism we implemented to target either high precision or high recall.

In all, the problem of novelty detection is deceptively difficult. We were struck by the difficulty that all groups in the Novelty Track in 2002 and 2003 had in obtaining high precision scores. Submissions that classify a very large proportion of the input sentences as novel reached the highest F-measure scores by getting high recall scores, but failed to achieve any substantial compression of material for users. Given that our goal is to generate an update summary, we focused on improving precision and increasing compression, removing as many false positives as possible.

The next section discusses the Novelty Track and the approaches others have tried; Section 3 details our system, and Section 4 presents the experiments.

2 Novelty Track

Much of the work in new-information detection has been done for the TREC Novelty Track. The task is related to first story detection, which is defined on whole documents rather than on passages within documents. In Task 1 of the Novelty Track, a system is given about 25 documents on a topic and asked to find all sentences relevant to the topic. In Task 2, the inputs are the set of relevant sentences, so that the program does not see the entire documents. The program must scan the sentences in order and output all that contain new information, that is information not seen in the previous input sentences.

2.1 Related Work

At the recent TREC, Dublin City University did well by comparing the words in a sentence against the accumulated words in all previous sentences (Blott et al., 2004). Their runs varied the way in which the words were weighted with frequency and inverse document frequency. Like our system, theirs follows from the intuition that words that are new to a discussion are evidence of novelty. But our system distinguishes between several kinds of words, including common nouns, named persons, named organization, etc. Our system also incorporates a mechanism for looking at the context of the sentence.

Both the Dublin system and ours are preceded by the University of Iowa's approach at TREC 2003. It based novelty decisions on a straightforward count of new named entities and noun phrases in a sentence (Eichmann et al., 2003). In 2004, the Iowa system (Eichmann et al., 2004) tried several embellishments, one using synonyms in addition to the words for novelty comparisons, and one using word-sense disambiguation. These two runs were above average in F-measure and about average in precision.

The University of Massachusetts system (Abdul-Jaleel et al., 2004) mixed a vector-space model with cosine similarity and a count of previously unseen named entities. Their system resembled one of two baseline methods that we submitted without our focus feature. Their submission used a similarity threshold that was tuned experimentally, while ours was learned automatically. In earlier work with the TREC 2002 data, UMass (Allan et al., 2003) compared a number of sentence-based models ranging in complexity from a count of new words and cosine distance, to a variety of sophisticated models based on KL divergence with different smoothing strategies and a "core mixture model" that considered the distribution of the words in the sentence with the distributions in a topic model and a general English model.

A number of groups have experimented with matrix-based methods. In 2003, a group from the University of Maryland and the Center for Computing Sciences (Conroy et al., 2003) used three techniques that used QR decomposition and singular value decomposition. The University of Maryland, Baltimore County, worked with clustering algorithms and singular value decomposition in sentence-sentence similarity matrices (Kallurkar et al., 2003). In 2004, Conroy (Conroy, 2004) tested Maximal Marginal Relevance (Goldstein et al., 2000) as well as QR decomposition.

The information retrieval group at Tsinghua University used a pooling technique, grouping similar sentences into clusters in order to capture sentences that partially match two or more other sentences(Ru et al., 2004). They said they had found difficulties

[1]Task 4 was similar to Task 2, in that both have the human annotations as input. For Task 2, participants only get the annotations, but in Task 4, they also receive the novel sentences from the fi rst fi ve documents as input. We felt that we would learn as much from the one task as from both.

with sentence-by-sentence comparisons.

2.2 Precision

At all three Novelty Track evaluations, from 2002 to 2004, it is clear that high precision is much harder to obtain than high recall. Trivial baselines – such as accept all sentences as novel – have proved to be difficult to beat by very much. This one-line algorithm automatically obtains 100% recall and precision equal to the proportion of novel sentences in the input. In 2003, when 66% of the relevant sentences were novel, the mean precision score was 0.635[2] and the median was 0.7. In 2004, 41% of the relevant sentences were novel, and the average precision dropped to 0.46. The median precision was also 0.46. Meanwhile, average recall scores across all submissions actually rose to 0.861 in 2004, compared with 0.795 in 2003. In terms of a real world system, this means that as the number of target sentences shrank, the number of sentences in the average program output rose. Likewise, a trivial system could guarantee no errors by returning nothing, but this would have no value.

2.3 Sentences

Normally, in Information Retrieval tasks, stricter thresholds result in higher precision, and looser thresholds, higher recall. In that way, a system can target its results to a user's needs. But in new-information detection, this rule of thumb fails at some point as thresholds become stricter. Recall does fall, but precision does not rise. In other words, there seems to be a ceiling for precision.

Several participants noted that their simpler strategies produced the best results. For example, in 2003, the Chinese Academy of Sciences (Sun et al., 2003), noted that word overlap was surprisingly strong as a similarity measure. As we have seen above, the Iowa approach of counting nouns was incorporated by a few others for 2004, including us. This strategy compares words in a sentence against all previous seen words and thus, avoids computing pairwise similarity between all sentences. Almost all participants performed such pairwise comparisons of systems.

A sentence-by-sentence comparison is clearly not the optimal operation for establishing novelty. Sentences with a large amount of overlap can express very different thoughts. In the extreme, a single word change can reverse the meaning of two sentences: *accept* and *reject*. This phenomenon led the Tsinghua University group to remark, "many sentences with an overlap of nearly 1 are real novel ones." (Ru et al., 2004).

On the other hand, it's not hard to find cases where realizations of equivalent statements take many different surface forms – with different choices of words and different syntactic structures. The data in the Novelty Task is drawn from three news services and clustered into fairly cohesive sets. The news writers consciously try to avoid echoing each other, and over time, echoing themselves. Sentences such as these have low word overlap, but are not novel. For this reason, we turned to a strategy of classifying each sentence S_i against the cumulative background of all the words in all preceding sentences $S_{1 \ldots i-1}$.

3 System

The system described in this paper was built with the Novelty Track in mind. The goal was to look at ways to consider longer spans of text than a sentence, and to avoid sentence by sentence comparisons.

In the Novelty track, the relevant sentences are presented in natural order, i.e. by the date of the document they came from, and then by their location in the document. The key characteristics of our program include:

- For each relevant sentence, our program calculates a sum of novel terms, which are terms that have not been previously seen. The terms are weighted according to their category, like person, location, common noun or verb. The weights are learned automatically.

- For the entire set, the program maintains a focus variable, which indicates whether the previous sentence is novel or old. Thresholds determine whether to continue or shift the focus. These are also learned automatically.

[2]One group appeared to have submitted a large number of irrelevant sentences in its submission, since it obtained relatively high recall scores, but very low precision scores, causing the average to drop below 0.66. The average precision of all other groups is about 0.7.

All input documents are fed in parallel into a named-entity recognizer, which marks persons, organizations, locations, part-of-speech tags for common nouns, and into a finite-state parser, which is used only to identify sentences beginning with subject pronouns. The output from the two preprocessing modules are merged and sent to the classifier.

The classifier reads a configuration file that contains a set of weights that were learned on the 2003 Novelty Track data to apply to different classes of words that have not been previously seen.

For each sentence, the system adds up the amount of novelty from the weighted terms in a sentence and compares that to a learned threshold; it classifies the sentence as novel if it exceeds the threshold. It also stores the classification in a focus variable. If the novelty threshold is not met, the system performs a series of tests described below, and possibly classifies some sentences with few content words as novel, depending on the status of the focus variable. Our algorithm enumerates all cases of changes in focus, and tests these in the order that allows the system to make the decision it can be most confident about first. Thus, when we find a named entity new to the discussion, we can be pretty sure that we have found a novel sentence. We can classify that sentence as new without regard to what preceded it. But, when we find a sentence devoid of high-content words, like "She said the idea sounded good," the system uses the classification of the previous sentence. If the antecedents to *she* or *idea* are novel, then this sentence must also be novel. The series of learned thresholds are imposed in a cascade to maximize the number of correct decisions over the training cases, in hopes the values will also cover unseen cases.

Thus, the classifier puts each sentence through the tests below, using the learned thresholds and weights described in Section 3.1. If any test succeeds, the system goes on to the next sentence.

1. *If there is a sufficient concentration of novel words, classify the sentence as novel* A sufficient concentration occurs when the sum of the weights of the novel content words (including named entities) exceeds a threshold, T_{novel}. If the previous focus was old, this indicates the focus has shifted to a novel segment.

2. *If there is a lack of novel words, classify the sentence as old* This is computed by comparing the sum of the weights of the already-seen content words to a separate threshold, T_{old}. If the previous focus was novel, this means the focus has shifted to an old segment.

3. For any remaining sentences, the classification is based on context:

   (a) *If the sentence does not have a sufficient number of content words, use the classification in the focus variable.* This adds the sums of both new and old content words and compares that to a threshold, T_{keep}.

   (b) *If the first noun phrase is a third person personal pronoun, use the classification in the focus variable.* Pronouns are known to signal that the same focus continues (Grosz and Sidner, 1986).

   (c) *If the sentence has not met any of the above tests but has a minimum number of content words, shift the focus.* If all tests above fail and there are a minimum number of content words, with a sum of T_{shift} shift the focus.

4. *Default* This rarely occurs but the default is to continue the focus, whether novel or old.

We examined the 2003 Novelty Track data and found that more than half the novel sentences appear in sequences of consecutive sentences (See Table 1). This circumstance creates an opportunity to make principled classifications on some sentences that have few, if any, clearly novel words, but continue a new segment. The use of a focus variable handles these cases.

3.1 Learning

In all, the system uses 11 real-valued parameters, weights and thresholds, and we wanted to learn optimal values for these. In particular, we wanted to be able to target either high recall or high precision, As we noted above, precision was much more difficult, and for a summarization task, much more important.

To learn the optimal values for the parameters, we opted to use an ad hoc algorithm. The main advantage in doing so was when considering instance i, the program can reference the classifications made

Length of Run	Count
1	1338
2	421
3	132
4	72
5	43
6	22
7	11
8	2
9	3
10	3
11	2
12	2
15	2
17	1

Table 1: Novelty often comes in bursts. This table shows that 1,338 of the novel sentences in the 2003 evaluation were singletons, and not a part of a run of novel sentences. Meanwhile, 1,526 of the sentences were part of runs of 2, 3 or 4 sentences.

for instance $i-1$, $i-2$, and possibly all the way back to instance 1, because the classification for instance i partly depends on the classification of previous instances. Not only do many standard supervised learning methods assume conditional independence, but they also do not provide access to the online classifications during learning. We constructed a randomized hill-climbing. The learner is structured like a neural net, but the weight adjustments are chosen at random as they are in genetic algorithms (See Figure 1). The evaluation, or fitness function, is the Novelty Track score itself, and the training data was the 2003 Novelty Track data.

Changes to the hypothesis are selected at random and evaluated. If the change does not hurt results, it is accepted. Otherwise the program backtracks and chooses another weight to update. We required the new configuration to produce a score greater than or equal to the previous one before we accepted it. The choice of which weight to update is made at random, in an effort to avoid local minima in the search space, but with an important restriction: the previous n choices are kept in a history list, which is checked to avoid re-use. This list is updated at each iteration. The configurations usually converge well within 100 iterations.

```
1. Initialize weights, history
       Weights take random values
2. Run the system using current weight set
3. If current score $>=$ previous best
       Update previous best
4. Otherwise
       Undo move
5. Update history
6. Choose next weight to change
7. Go to step 2
```

Figure 1: The learning algorithm uses a randomized hill climbing approach with backtracking

3.2 Bias Adjustment

In training on the 2003 data, the biggest problem was to find a way to deal with the large percentage of novel sentences. About 65% of the instances are positive, so that a random system achieves a relatively high F-measure by increasing the number of sentences it calls novel – until recall reaches 1.0. Another strategy would be to choose only the sentences in the first document, achieving a high precision – more than 90% of the relevant sentences in the first document for each topic were called novel.

In the Novelty Track the F-measure was set to give equal weight to precision and recall, but we wanted to be able to coax the learner to give greater weight to either precision or by adjusting the F-measure computation:

$$F = \frac{1}{\frac{\beta}{prec} + \frac{(1-\beta)}{recall}}$$

β is a number between 0 and 1. The closer it gets to 1, the more the formula favors precision.

We chose whether to emphasize precision or recall by altering the value of β. At the most extreme, we set β at 0.9 for the largest emphasis on precision. When emphasizing recall, we left β at 0.5.

The design was motivated by the need to explore the problem more fully and inform the algorithm for deciding novelty as much as to find optimal parameters for the values. Thus, we wanted to be able to record all the steps the learner made through the search space, and to save the intermediate states. At times, the learner would settle into a configuration that produced a trivial solution, and we could choose one of the intermediate configurations that produced a more reasonable score.

3.3 Vector-Space Module

In addition to the system which integrates novel word features with focus tracking, we also implemented a vector-space approach as a baseline – the *Cosine* run. We tested the vector-space system alone to contrast it with the focus system, but we also tested a version which integrated the vector-space system with the focus system.

Our vector-space module assigns all non-stop-words a value of 1, and uses the cosine distance metric to compute similarity.

$$Cos(u,v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$$

and

$$Novel(s_i) \begin{cases} True & \text{if} \quad Cos(s_i, s_j) < T, \\ & \text{for} \quad j = 1 \ldots i-1 \\ False & \text{otherwise} \end{cases}$$

As each sentence is scanned, its similarity is computed with all previous sentences and the maximum similarity is compared to a threshold T. If that maximum exceeds T, it is considered novel. We chose the value of T after trials on the 2003 Novelty Track data. It was set at 0.385, resulting in a balanced system that matched the results of one of the strongest performers at the TREC evaluations that year.

On the 2003 data, when we set T at .9, we found that we had a precision of .71 and a recall of 0.98, indicating that about 6% of the sentences were quite similar to some preceding sentence (See Figure 2). After that, each point of precision was very costly in terms of recall. Our experience was mirrored by the participants at TREC 2003 and again at TREC 2004.

We considered this vector-space model to be our baseline. We also tried it in combination with the *Recall* run explained above. Because both the *Recall* and *Cosine* runs produced a relatively large output and because they used different methods, we thought the intersection would result in higher precision, though with some loss of recall.

In practice, the range of recall was much greater than precision. Judging from the experiences of the participants at TREC and our own exploratory experiments, it was difficult to push precision above 0.80 with the TREC 2003 data, and above 0.50 with the TREC 2004 data.

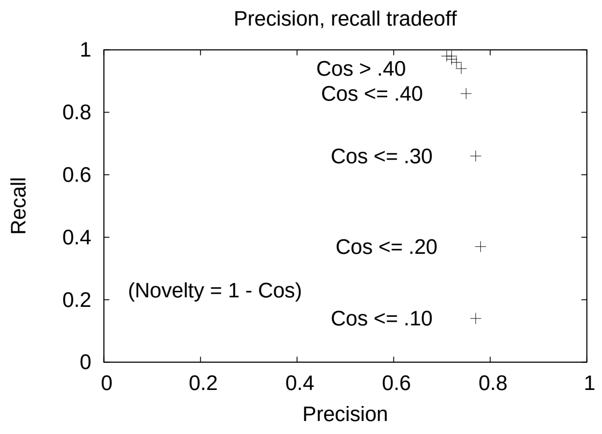


Figure 2: The precision and recall scores of a vector-space model with cosine similarity at different thresholds, on the TREC 2003 data. Making the test for novelty stricter fails to improve precision but has a drastic effect on recall.

4 Experiments

4.1 Results from TREC 2004

Our results are encouraging, especially since the configurations that were oriented toward higher precision, indeed, achieved the best precision scores in the evaluation, with our best precision run about 20% higher in precision than the best of all the runs by other groups (See Figure 3.) Meanwhile, our recall-oriented run was one of eight runs that were in a virtual tie for achieving the top f-measure. These eight runs were within 0.01 of one another in the measure.

Our five submitted runs were:

Prec1 aimed at moderately high precision, with reasonable recall.

Prec2 aimed at high precision, with little attention to recall.

Recall weighted precision and recall equally.

Cosine a baseline of a standard vector-space model with a cosine similarity metric.

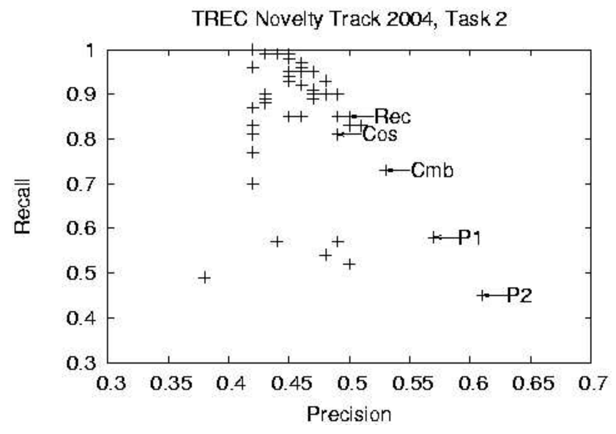


Figure 3: The graph shows all 54 submission in Task 2 for the Novelty Track, with our five submissions labeled. Our precision-oriented runs were well ahead of all others in precision, while our recall-oriented run was in a large group that reached about 0.5 precision with relatively high recall.

Combo a composite submission using the intersection of Recall and Cosine.

Table 2 shows the numbers of our performance of our five submissions. *Prec1* had an F-score close to the average of 0.577 for all systems, while *Prec2* was 50% ahead of random selection in accuracy. Both our *Combo* system and our baseline *Cosine* were above average in F-measure. Our emphasis on precision is justified in a number of ways, although the official yardstick was the F-measure.

An analysis of the system's behavior under the different parameters showed that the precision-oriented runs, in particular *Prec1*, valued verbs and common nouns more than named entities in deciding novelty. The precision-oriented runs also benefited more from the focus variable, with their scores about 5% higher in terms of F-measure than they were without it. The pronoun test, however, was rarely used, firing less than 1% of the time.

We note that we are developing novelty detection for summarization, where compression of the report is valuable. Table 2 shows the lengths of our returns. It is impossible to compare these precisely with other systems, because the averages given by NIST are averages of the scores for each of the 50 sets, and we do not have the breakdown of the numbers by set for any submissions but our own. However, we can estimate the size of the other output by considering average precision and recall as if they were computed over the total number of sentences in all 50 sets. This computation shows an average output for all participants of about 6,500 sentences and a median of 6,981 – out of a total of 8,343 sentences. However, this total includes some amount of header material, not only the headline, but the document ID and other identifiers, the date and some shorthand messages from the wire services to its clients. In addition, a number of the sets had near perfect duplicate articles. This is in sharp contrast with typical summaries. At the 2004 Document Understanding Conference, the typical input cluster contained more than 4,000 words, and the task required that this be reduced to 100 words. We contend there is little value in a system that does no more than weed out very few sentences, even though they might have achieved high F-measures.

Second, our experience, and the results of other groups, shows that high precision is harder than high recall. In all three years of the Novelty Track, precision scores tended to hover in a narrow band just above what one would get by mechanically labeling all sentences as *novel*.

5 Conclusion

The success of our use of context in the TREC Novelty Track led us to incorporate the idea into a larger system. This system identifies *clauses* within sentences that express new information and tries to identify semantic equivalents. It is being developed as part of a multi-document summarizer that produces topical updates for users.

In addition, the work here suggests three directions for future work:

- Adapt the features used here to some of the newer probabilistic formalisms, like conditional random fields.
- Try full segmentation of the input documents rather than treat the sentences as a sequence.
- Try to identify all nominal references to canonical forms.

With this experimental system, we obtained the the top precision scores in the Novelty Track, and

Run-Id	Precision	Recall	F-meas	Output length
Prec1	0.57	0.58	0.562	3276
Prec2	0.61	0.45	0.506	2372
Recall	0.51	0.82	0.611	5603
Cosine	0.49	0.81	0.599	5537
Combo	0.53	0.73	0.598	4578
Choose All	0.41	1.000	0.581	8343
Average All Runs	0.46	0.86	0.577	6500

Table 2: Comparison of results of our five runs, compared to a random selection of sentences, and the overall average F-scores by all 55 submissions.

we obtained the program settings to do this automatically. High precision is very difficult to obtain, and every point in precision costs too much in recall. Further exploration is needed to determine whether linguistic knowledge will help, and whether state-of-the-art tools are powerful enough to improve performance.

Beyond new-information detection, the idea of tracking context with a surface means like the focus variable is worth exploring in other tasks, including summarization and question-answering.

References

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. Umass at trec 2004: Notebook. In *The Thirteenth Text Retrieval Conference (TREC 2004) Notebook*.

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of the ACM SIGIR conference on research and development in information retrieval*.

Stephen Blott, Oisin Boydell, Fabrice Camous, Paul Ferguson, Georgina Gaughan, Cathal Gurrin, Noel Murphy, Noel O'Connor, Alan F. Smeaton, Barry Smyth, and Peter Wilkins. 2004. Experiments in terabyte searching, genomic retrieval and novelty detection for trec-2004. In *The Thirteenth Text Retrieval Conference (TREC 2004) Notebook*.

John M Conroy, Daniel M. Dunlavy, and Dianne P. O'Leary. 2003. From trec to duc to trec again. In *TREC Notebook Proceedings*.

John M. Conroy. 2004. A hidden markov model for trec's novelty task. In *The Thirteenth Text Retrieval Conference (TREC 2004) Notebook*.

David Eichmann, Padmini Srinivasan, Marc Light, Hudong Wang, Xin Ying Qiu, Robert J. Arens, and Aditya Sehgal. 2003. Experiments in novelty, genes and questions at the university of iowa. In *TREC Notebook Proceedings*.

David Eichmann, Yi Zhang, Shannon Bradshaw, Xin Ying Qiu, Li Zhou, Padmini Srinivasan, Aditya Kumar Sehgal, and Hudon Wong. 2004. Novelty, question answering and genomics: The university of iowa response. In *The Thirteenth Text Retrieval Conference (TREC 2004) Notebook*.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of ANLP/NAACL-2000 Workshop on Automatic Summarization*.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Srikanth Kallurkar, Yongmei Shi, R. Scott Cost, Charles Nicholas, Akshay Java, Christopher James, Sowjanya Rajavaram, Vishal Shanbhag, Sachin Bhatkar, and Drew Ogle. 2003. Umbc at trec 12. In *TREC Notebook Proceedings*.

R. Ohgaya, A. Shimmura, and T. Takagi. 2003. Meiji university web and novelty track experiments at trec 2003. In *TREC Notebook Proceedings*.

Liyun Ru, Le Zhao, Min Zhang, and Shaoping Ma. 2004. Improved feature selection and redundancy computing – thuir at trec 2004 novelty track. In *The Thirteenth Text Retrieval Conference (TREC 2004) Notebook*.

Ian Soboroff. 2004. Draft overview of the trec 2004 novelty track. In *The Thirteenth Text Retrieval Conference (TREC 2004) Notebook*.

Jian Sun, Wenfeng pan, Huaping Zhang, Zhe Yang, Bin Wang, Gang Zhang, and Xueqi Cheng. 2003. Trec-2003 novelty and web track at ict. In *TREC Notebook Proceedings*.

A Shortest Path Dependency Kernel for Relation Extraction

Razvan C. Bunescu and **Raymond J. Mooney**
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
razvan,mooney@cs.utexas.edu

Abstract

We present a novel approach to relation extraction, based on the observation that the information required to assert a relationship between two named entities in the same sentence is typically captured by the shortest path between the two entities in the dependency graph. Experiments on extracting top-level relations from the ACE (Automated Content Extraction) newspaper corpus show that the new shortest path dependency kernel outperforms a recent approach based on dependency tree kernels.

1 Introduction

One of the key tasks in natural language processing is that of Information Extraction (IE), which is traditionally divided into three subproblems: coreference resolution, named entity recognition, and relation extraction. Consequently, IE corpora are typically annotated with information corresponding to these subtasks (MUC (Grishman, 1995), ACE (NIST, 2000)), facilitating the development of systems that target only one or a subset of the three problems. In this paper we focus exclusively on extracting relations between predefined types of entities in the ACE corpus. Reliably extracting relations between entities in natural-language documents is still a difficult, unsolved problem, whose inherent difficulty is compounded by the emergence of new application domains, with new types of narrative that challenge systems developed for previous well-studied domains. The accuracy level of current syntactic and semantic parsers on natural language text from different domains limit the extent to which syntactic and semantic information can be used in real IE systems. Nevertheless, various lines of work on relation extraction have shown experimentally that the use of automatically derived syntactic information can lead to significant improvements in extraction accuracy. The amount of syntactic knowledge used in IE systems varies from part-of-speech only (Ray and Craven, 2001) to chunking (Ray and Craven, 2001) to shallow parse trees (Zelenko et al., 2003) to dependency trees derived from full parse trees (Culotta and Sorensen, 2004). Even though exhaustive experiments comparing the performance of a relation extraction system based on these four levels of syntactic information are yet to be conducted, a reasonable assumption is that the extraction accuracy increases with the amount of syntactic information used. The performance however depends not only on the amount of syntactic information, but also on the details of the exact models using this information. Training a machine learning system in a setting where the information used for representing the examples is only partially relevant to the actual task often leads to overfitting. It is therefore important to design the IE system so that the input data is stripped of unnecessary features as much as possible. In the case of the tree kernels from (Zelenko et al., 2003; Culotta and Sorensen, 2004), the authors reduce each relation example to the smallest subtree in the parse or dependency tree that includes both entities. We will show in this paper that increased extraction performance can be

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 724–731, Vancouver, October 2005. ©2005 Association for Computational Linguistics

obtained by designing a kernel method that uses an even smaller part of the dependency structure – the shortest path between the two entities in the undirected version of the dependency graph.

2 Dependency Graphs

Let e_1 and e_2 be two entities mentioned in the same sentence such that they are observed to be in a relationship R, i.e $R(e_1, e_2) = 1$. For example, R can specify that entity e_1 is LOCATED (AT) entity e_2. Figure 1 shows two sample sentences from ACE, with entity mentions in bold. Correspondingly, the first column in Table 1 lists the four relations of type LOCATED that need to be extracted by the IE system. We assume that a relation is to be extracted only between entities mentioned in the same sentence, and that the presence or absence of a relation is independent of the text preceding or following the sentence. This means that only information derived from the sentence including the two entities will be relevant for relation extraction. Furthermore, with each sentence we associate its dependency graph, with words figured as nodes and word-word dependencies figured as directed edges, as shown in Figure 1. A subset of these word-word dependencies capture the predicate-argument relations present in the sentence. Arguments are connected to their target predicates either directly through an arc pointing to the predicate ('troops $\rightarrow$ raided'), or indirectly through a preposition or infinitive particle ('warning $\leftarrow$ to $\leftarrow$ stop'). Other types of word-word dependencies account for modifier-head relationships present in adjective-noun compounds ('several $\rightarrow$ stations'), noun-noun compounds ('pumping $\rightarrow$ stations'), or adverb-verb constructions ('recently $\rightarrow$ raided'). In Figure 1 we show the full dependency graphs for two sentences from the ACE newspaper corpus.

Word-word dependencies are typically categorized in two classes as follows:

- **[Local Dependencies]** These correspond to local predicate-argument (or head-modifier) constructions such as 'troops $\rightarrow$ raided', or 'pumping $\rightarrow$ stations' in Figure 1.

- **[Non-local Dependencies]** Long-distance dependencies arise due to various linguistic constructions such as coordination, extraction, raising and control. In Figure 1, among non-local dependencies are 'troops $\rightarrow$ warning', or 'ministers $\rightarrow$ preaching'.

A Context Free Grammar (CFG) parser can be used to extract local dependencies, which for each sentence form a dependency tree. Mildly context sensitive formalisms such as Combinatory Categorial Grammar (CCG) (Steedman, 2000) model word-word dependencies more directly and can be used to extract both local and long-distance dependencies, giving rise to a directed acyclic graph, as illustrated in Figure 1.

3 The Shortest Path Hypothesis

If e_1 and e_2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, our hypothesis stipulates that the contribution of the sentence dependency graph to establishing the relationship $R(e_1, e_2)$ is almost exclusively concentrated in the shortest path between e_1 and e_2 in the undirected version of the dependency graph.

If entities e_1 and e_2 are arguments of the same predicate, then the shortest path between them will pass through the predicate, which may be connected directly to the two entities, or indirectly through prepositions. If e_1 and e_2 belong to different predicate-argument structures that share a common argument, then the shortest path will pass through this argument. This is the case with the shortest path between 'stations' and 'workers' in Figure 1, passing through 'protesters', which is an argument common to both predicates 'holding' and 'seized'. In Table 1 we show the paths corresponding to the four relation instances encoded in the ACE corpus for the two sentences from Figure 1. All these paths support the LOCATED relationship. For the first path, it is reasonable to infer that if a PERSON entity (e.g. 'protesters') is doing some action (e.g. 'seized') to a FACILITY entity (e.g. 'station'), then the PERSON entity is LOCATED at that FACILITY entity. The second path captures the fact that the same PERSON entity (e.g. 'protesters') is doing two actions (e.g. 'holding' and 'seized') , one action to a PERSON entity (e.g. 'workers'), and the other action to a FACILITY entity (e.g. 'station'). A reasonable inference in this case is that the 'workers' are LOCATED at the

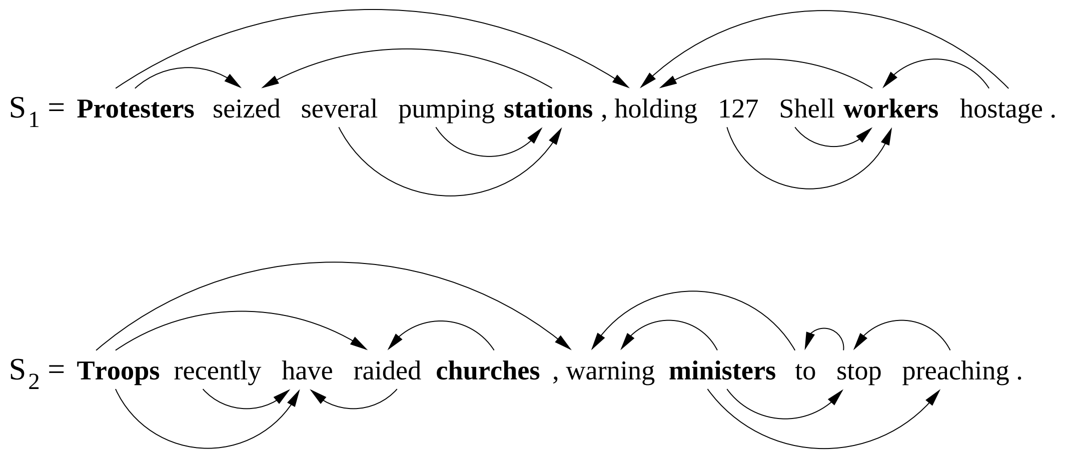

S 1 = Protesters seized several pumping stations , holding 127 Shell workers hostage .
S 2 = Troops recently have raided churches , warning ministers to stop preaching .

Figure 1: Sentences as dependency graphs.

Relation Instance	Shortest Path in Undirected Dependency Graph
S_1: protesters AT stations	**protesters** $\longrightarrow$ seized $\longleftarrow$ **stations**
S_1: workers AT stations	**workers** $\longrightarrow$ holding $\longleftarrow$ protesters $\longrightarrow$ seized $\longleftarrow$ **stations**
S_2: troops AT churches	**troops** $\longrightarrow$ raided $\longleftarrow$ **churches**
S_2: ministers AT churches	**ministers** $\longrightarrow$ warning $\longleftarrow$ troops $\longrightarrow$ raided $\longleftarrow$ **churches**

Table 1: Shortest Path representation of relations.

’station’.

In Figure 2 we show three more examples of the LOCATED (AT) relationship as dependency paths created from one or two predicate-argument structures. The second example is an interesting case, as it illustrates how annotation decisions are accommodated in our approach. Using a reasoning similar with that from the previous paragraph, it is reasonable to infer that ’troops’ are LOCATED in ’vans’, and that ’vans’ are LOCATED in ’city’. However, because ’vans’ is not an ACE markable, it cannot participate in an annotated relationship. Therefore, ’troops’ is annotated as being LOCATED in ’city’, which makes sense due to the transitivity of the relation LOCATED. In our approach, this leads to shortest paths that pass through two or more predicate-argument structures.

The last relation example is a case where there exist multiple shortest paths in the dependency graph between the same two entities – there are actually two different paths, with each path replicated into three similar paths due to coordination. Our current approach considers only one of the shortest paths, nevertheless it seems reasonable to investigate using all of them as multiple sources of evidence for relation extraction.

There may be cases where e_1 and e_2 belong to predicate-argument structures that have no argument in common. However, because the dependency graph is always connected, we are guaranteed to find a shortest path between the two entities. In general, we shall find a shortest sequence of predicate-argument structures with target predicates $P_1, P_2, ..., P_n$ such that e_1 is an argument of P_1, e_2 is an argument of P_n, and any two consecutive predicates P_i and P_{i+1} share a common argument (where by “argument” we mean both arguments and complements).

4 Learning with Dependency Paths

The shortest path between two entities in a dependency graph offers a very condensed representation of the information needed to assess their relationship. A dependency path is represented as a sequence of words interspersed with arrows that in-

(1) He had no regrets for **his** actions in **Brcko**.

his → actions ← in ← **Brcko**

(2) U.S. **troops** today acted for the first time to capture an alleged Bosnian war criminal, rushing from unmarked vans parked in the northern Serb-dominated **city** of Bijeljina.

troops → rushing ← from ← vans → parked ← in ← **city**

(3) Jelisic created an atmosphere of terror at the **camp** by killing, abusing and threatening the **detainees**.

detainees → killing ← Jelisic → created ← at ← **camp**

detainees → abusing ← Jelisic → created ← at ← **camp**

detainees → threatning ← Jelisic → created ← at ← **camp**

detainees → killing → by → created ← at ← **camp**

detainees → abusing → by → created ← at ← **camp**

detainees → threatening → by → created ← at ← **camp**

Figure 2: Relation examples.

dicate the orientation of each dependency, as illustrated in Table 1. These paths however are completely lexicalized and consequently their performance will be limited by data sparsity. We can alleviate this by categorizing words into classes with varying degrees of generality, and then allowing paths to use both words and their classes. Examples of word classes are part-of-speech (POS) tags and generalizations over POS tags such as Noun, Active Verb or Passive Verb. The entity type is also used for the two ends of the dependency path. Other potentially useful classes might be created by associating with each noun or verb a set of hypernyms corresponding to their synsets in WordNet.

The set of features can then be defined as a Cartesian product over these word classes, as illustrated in Figure 3 for the dependency path between ’protesters’ and ’station’ in sentence S_1. In this representation, sparse or contiguous subsequences of nodes along the lexicalized dependency path (i.e. path fragments) are included as features simply by replacing the rest of the nodes with their corresponding generalizations.

The total number of features generated by this dependency path is $4 \times 1 \times 3 \times 1 \times 4$, and some of them are listed in Table 2.

$$\begin{bmatrix} \text{protesters} \\ \text{NNS} \\ \text{Noun} \\ \text{PERSON} \end{bmatrix} \times [\rightarrow] \times \begin{bmatrix} \text{seized} \\ \text{VBD} \\ \text{Verb} \end{bmatrix} \times [\leftarrow] \times \begin{bmatrix} \text{stations} \\ \text{NNS} \\ \text{Noun} \\ \text{FACILITY} \end{bmatrix}$$

Figure 3: Feature generation from dependency path.

protesters	→	seized	←	stations
Noun	→	Verb	←	Noun
PERSON	→	seized	←	FACILITY
PERSON	→	Verb	←	FACILITY
... (48 features)				

Table 2: Sample Features.

For verbs and nouns (and their respective word classes) occurring along a dependency path we also use an additional suffix ’(-)’ to indicate a negative polarity item. In the case of verbs, this suffix is used when the verb (or an attached auxiliary) is modified by a negative polarity adverb such as ’not’ or ’never’. Nouns get the negative suffix whenever they are modified by negative determiners such as ’no’, ’neither’ or ’nor’. For example, the phrase “**He** never went to **Paris**” is associated with the dependency path ’**He** → went(-) ← to ← **Paris**’.

Explicitly creating for each relation example a vector with a position for each dependency path feature is infeasible, due to the high dimensionality of the feature space. Here we can exploit *dual* learning algorithms that process examples only via computing their dot-products, such as the Support Vector Machines (SVMs) (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000). These dot-products between feature vectors can be efficiently computed through a *kernel* function, without iterating over all the corresponding features. Given the kernel function, the SVM learner tries to find a hyperplane that separates positive from negative examples and at the same time maximizes the separation (margin) between them. This type of max-margin separator has been shown both theoretically and empirically to resist overfitting and to provide good generalization performance on unseen examples.

Computing the dot-product (i.e. kernel) between two relation examples amounts to calculating the number of common features of the type illustrated in Table 2. If $\boldsymbol{x} = x_1 x_2 ... x_m$ and $\boldsymbol{y} = y_1 y_2 ... y_n$ are two relation examples, where x_i denotes the set of word classes corresponding to position i (as in Figure 3), then the number of common features between $\boldsymbol{x}$ and $\boldsymbol{y}$ is computed as in Equation 1.

$$K(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} 0, & m \neq n \\ \prod_{i=1}^{n} c(x_i, y_i), & m = n \end{cases} \tag{1}$$

where $c(x_i, y_i) = |x_i \cap y_i|$ is the number of common word classes between x_i and y_i.

This is a simple kernel, whose computation takes $O(n)$ time. If the two paths have different lengths, they correspond to different ways of expressing a relationship – for instance, they may pass through a different number of predicate argument structures. Consequently, the kernel is defined to be 0 in this case. Otherwise, it is the product of the number of common word classes at each position in the two paths. As an example, let us consider two instances of the LOCATED relationship:

1. '**his** actions in **Brcko**', and
2. '**his** arrival in **Beijing**'.

Their corresponding dependency paths are:

1. '**his** $\rightarrow$ actions $\leftarrow$ in $\leftarrow$ **Brcko**', and
2. '**his** $\rightarrow$ arrival $\leftarrow$ in $\leftarrow$ **Beijing**'.

Their representation as a sequence of sets of word classes is given by:

1. $\boldsymbol{x} = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7]$, where x_1 = {his, PRP, PERSON}, x_2 = {$\rightarrow$}, x_3 = {actions, NNS, Noun}, x_4 = {$\leftarrow$}, x_5 = {in, IN}, x_6 = {$\leftarrow$}, x_7 = {Brcko, NNP, Noun, LOCATION}
2. $\boldsymbol{y} = [y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7]$, where y_1 = {his, PRP, PERSON}, y_2 = {$\rightarrow$}, y_3 = {arrival, NN, Noun}, y_4 = {$\leftarrow$}, y_5 = {in, IN}, y_6 = {$\leftarrow$}, y_7 = {Beijing, NNP, Noun, LOCATION}

Based on the formula from Equation 1, the kernel is computed as $K(\boldsymbol{x},\boldsymbol{y}) = 3 \times 1 \times 1 \times 1 \times 2 \times 1 \times 3 = 18$.

We use this relation kernel in conjunction with SVMs in order to find decision hyperplanes that best separate positive examples from negative examples. We modified the LibSVM[1] package for SVM learning by plugging in the kernel described above, and used its default one-against-one implementation for multiclass classification.

5 Experimental Evaluation

We applied the shortest path dependency kernel to the problem of extracting top-level relations from the ACE corpus (NIST, 2000), the version used for the September 2002 evaluation. The training part of this dataset consists of 422 documents, with a separate set of 97 documents allocated for testing. This version of the ACE corpus contains three types of annotations: coreference, named entities and relations. Entities can be of the type PERSON, ORGANIZATION, FACILITY, LOCATION, and GEO-POLITICAL ENTITY. There are 5 general, top-level relations: ROLE, PART, LOCATED, NEAR, and SOCIAL. The ROLE relation links people to an organization to which they belong, own, founded, or provide some service. The PART relation indicates subset relationships, such as a state to a nation, or a subsidiary to its parent company. The AT relation indicates the location of a person or organization at some location. The NEAR relation indicates the proximity of one location to another. The SOCIAL relation links two people in personal, familial or professional relationships. Each top-level relation type is further subdivided into more fine-grained subtypes, resulting in a total of 24 relation types. For example, the LOCATED relation includes subtypes such as LOCATED-AT, BASED-IN, and RESIDENCE. In total, there are 7,646 intra-sentential relations, of which 6,156 are in the training data and 1,490 in the test data.

We assume that the entities and their labels are known. All preprocessing steps – sentence segmentation, tokenization, and POS tagging – were performed using the OpenNLP[2] package.

5.1 Extracting dependencies using a CCG parser

CCG (Steedman, 2000) is a type-driven theory of grammar where most language-specific aspects of the grammar are specified into lexicon. To each lex-

[1]URL:http://www.csie.ntu.edu.tw/˜cjlin/libsvm/
[2]URL: http://opennlp.sourceforge.net

ical item corresponds a set of syntactic categories specifying its valency and the directionality of its arguments. For example, the words from the sentence “protesters seized several stations” are mapped in the lexicon to the following categories:

protesters	:	NP
seized	:	$(S\backslash NP)/NP$
several	:	NP/NP
stations	:	NP

The transitive verb ’seized’ expects two arguments: a noun phrase to the right (the object) and another noun phrase to the left (the subject). Similarly, the adjective ’several’ expects a noun phrase to its right. Depending on whether its valency is greater than zero or not, a syntactic category is called a *functor* or an *argument*. In the example above, ’seized’ and ’several’ are functors, while ’protesters’ and ’stations’ are arguments.

Syntactic categories are combined using a small set of typed combinatory rules such as functional application, composition and type raising. In Table 3 we show a sample derivation based on three functional applications.

protesters	seized	several	stations
NP	$(S\backslash NP)/NP$	NP/NP	NP
NP	$(S\backslash NP)/NP$	NP	
NP	$S\backslash NP$		
S			

Table 3: Sample derivation.

In order to obtain CCG derivations for all sentences in the ACE corpus, we used the CCG parser introduced in (Hockenmaier and Steedman, 2002)[3]. This parser also outputs a list of dependencies, with each dependency represented as a 4-tuple $\langle f, a, w_f, w_a\rangle$, where f is the syntactic category of the functor, a is the argument number, w_f is the head word of the functor, and w_a is the head word of the argument. For example, the three functional applications from Table 3 result in the functor-argument dependencies enumerated below in Table 4.

[3]URL:http://www.ircs.upenn.edu/˜juliahr/Parser/

f	a	w_f	w_a
NP/NP	1	’several’	’stations’
$(S\backslash NP)/NP$	2	’seized’	’stations’
$(S\backslash NP)/NP$	1	’seized’	’protesters’

Table 4: Sample dependencies.

Because predicates (e.g. ’seized’) and adjuncts (e.g. ’several’) are always represented as functors, while complements (e.g. ’protesters’ and ’stations’) are always represented as arguments, it is straightforward to transform a functor-argument dependency into a head-modifier dependency. The head-modifier dependencies corresponding to the three functor-argument dependencies in Table 4 are: ’protesters $\rightarrow$ seized’, ’stations $\rightarrow$ seized’, and ’several $\rightarrow$ stations’.

Special syntactic categories are assigned in CCG to lexical items that project unbounded dependencies, such as the relative pronouns ’who’, ’which’ and ’that’. Coupled with a head-passing mechanism, these categories allow the extraction of long-range dependencies. Together with the local word-word dependencies, they create a directed acyclic dependency graph for each parsed sentence, as shown in Figure 1.

5.2 Extracting dependencies using a CFG parser

Local dependencies can be extracted from a CFG parse tree using simple heuristic rules for finding the head child for each type of constituent. Alternatively, head-modifier dependencies can be directly output by a parser whose model is based on lexical dependencies. In our experiments, we used the full parse output from Collins’ parser (Collins, 1997), in which every non-terminal node is already annotated with head information. Because local dependencies assemble into a tree for each sentence, there is only one (shortest) path between any two entities in a dependency tree.

5.3 Experimental Results

A recent approach to extracting relations is described in (Culotta and Sorensen, 2004). The authors use a generalized version of the tree kernel from (Zelenko et al., 2003) to compute a kernel over relation examples, where a relation example consists of the smallest dependency tree containing the two entities of the relation. Precision and recall values are reported for the task of extracting the 5 top-level relations in the ACE corpus under two different scenarios:

– **[S1]** This is the classic setting: one multi-class SVM is learned to discriminate among the 5 top-level classes, plus one more class for the no-relation cases.

– **[S2]** Because of the highly skewed data distribution, the recall of the SVM approach in the first scenario is very low. In (Culotta and Sorensen, 2004) the authors propose doing relation extraction in two steps: first, one binary SVM is trained for *relation detection*, which means that all positive relation instances are combined into one class. Then the thresholded output of this binary classifier is used as training data for a second multi-class SVM, which is trained for *relation classification*. The same kernel is used in both stages.

We present in Table 5 the performance of our shortest path (SP) dependency kernel on the task of relation extraction from ACE, where the dependencies are extracted using either a CCG parser (SP-CCG), or a CFG parser (SP-CFG). We also show the results presented in (Culotta and Sorensen, 2004) for their best performing kernel K4 (a sum between a bag-of-words kernel and their dependency kernel) under both scenarios.

Method	Precision	Recall	F-measure
(S1) SP-CCG	67.5	37.2	48.0
(S1) SP-CFG	**71.1**	**39.2**	**50.5**
(S1) K4	70.3	26.3	38.0
(S2) SP-CCG	63.7	41.4	50.2
(S2) SP-CFG	**65.5**	**43.8**	**52.5**
(S2) K4	67.1	35.0	45.8

Table 5: Extraction Performance on ACE.

The shortest-path dependency kernels outperform the dependency kernel from (Culotta and Sorensen, 2004) in both scenarios, with a more significant difference for SP-CFG. An error analysis revealed that Collins' parser was better at capturing local dependencies, hence the increased accuracy of SP-CFG. Another advantage of our shortest-path dependency kernels is that their training and testing are very fast – this is due to representing the sentence as a chain of dependencies on which a fast kernel can be computed. All the four SP kernels from Table 5 take between 2 and 3 hours to train and test on a 2.6GHz Pentium IV machine.

To avoid numerical problems, we constrained the dependency paths to pass through at most 10 words (as observed in the training data) by setting the kernel to 0 for longer paths. We also tried the alternative solution of normalizing the kernel, however this led to a slight decrease in accuracy. Having longer paths give larger kernel scores in the unnormalized version does not pose a problem because, by definition, paths of different lengths correspond to disjoint sets of features. Consequently, the SVM algorithm will induce lower weights for features occurring in longer paths, resulting in a linear separator that works irrespective of the size of the dependency paths.

6 Related Work

In (Zelenko et al., 2003), the authors do relation extraction using a tree kernel defined over shallow parse tree representations of sentences. The same tree kernel is slightly generalized in (Culotta and Sorensen, 2004) and used in conjunction with dependency trees. In both approaches, a relation instance is defined to be the smallest subtree in the parse or dependency tree that includes both entities. In this paper we argued that the information relevant to relation extraction is almost entirely concentrated in the shortest path in the dependency tree, leading to an even smaller representation. Another difference between the tree kernels above and our new kernel is that the tree kernels used for relation extraction are *opaque* i.e. the semantics of the dimensions in the corresponding Hilbert space is not obvious. For the shortest-path kernels, the semantics is known by definition: each path feature corresponds to a dimension in the Hilbert space. This transparency allows us to easily restrict the types of patterns counted by the kernel to types that we deem relevant for relation extraction. The tree kernels are also more time consuming, especially in the sparse setting, where they count sparse subsequences of children common to nodes in the two trees. In (Zelenko et al., 2003), the tree kernel is computed in $O(mn)$ time, where m and n are the number of nodes in the two trees. This changes to $O(mn^3)$ in the sparse setting.

Our shortest-path intuition bears some similarity with the underlying assumption of the relational pathfinding algorithm from (Richards and Mooney, 1992) : "in most relational domains, important concepts will be represented by a small number of fixed paths among the constants defining a positive instance – for example, the grandparent relation is defined by a single fixed path consisting of two parent relations." We can see this happening also in the task of relation extraction from ACE, where "important concepts" are the 5 types of relations, and the "constants" defining a positive instance are the 5 types of entities.

7 Future Work

Local and non-local (deep) dependencies are equally important for finding relations. In this paper we tried extracting both types of dependencies using a CCG parser, however another approach is to recover deep dependencies from syntactic parses, as in (Campbell, 2004; Levy and Manning, 2004). This may have the advantage of preserving the quality of local dependencies while completing the representation with non-local dependencies.

Currently, the method assumes that the named entities are known. A natural extension is to automatically extract both the entities and their relationships. Recent research (Roth and Yih, 2004) indicates that integrating entity recognition with relation extraction in a global model that captures the mutual influences between the two tasks can lead to significant improvements in accuracy.

8 Conclusion

We have presented a new kernel for relation extraction based on the shortest-path between the two relation entities in the dependency graph. Comparative experiments on extracting top-level relations from the ACE corpus show significant improvements over a recent dependency tree kernel.

9 Acknowledgements

This work was supported by grants IIS-0117308 and IIS-0325116 from the NSF.

References

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 645–652, Barcelona, Spain, July.

Michael J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, pages 16–23.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, July.

Ralph Grishman. 1995. Message Understanding Conference 6. http://cs.nyu.edu/cs/faculty/grishman/muc6.html.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 335–342, Philadelphia, PA.

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 (ACL-04)*, pages 327–334, Barcelona, Spain, July.

NIST. 2000. ACE – Automatic Content Extraction. http://www.nist.gov/speech/tests/ace.

Soumya Ray and Mark Craven. 2001. Representing sentence structure in hidden Markov models for information extraction. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 1273–1279, Seattle, WA.

Bradley L. Richards and Raymond J. Mooney. 1992. Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 50–55, San Jose, CA, July.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8, Boston, MA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

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

Multi-way Relation Classification: Application to Protein-Protein Interactions

Barbara Rosario
SIMS
UC Berkeley
Berkeley, CA 94720
rosario@sims.berkeley.edu

Marti A. Hearst
SIMS
UC Berkeley
Berkeley, CA 94720
hearst@sims.berkeley.edu

Abstract

We address the problem of multi-way relation classification, applied to identification of the interactions between proteins in bioscience text. A major impediment to such work is the acquisition of appropriately labeled training data; for our experiments we have identified a database that serves as a proxy for training data. We use two graphical models and a neural net for the classification of the interactions, achieving an accuracy of 64% for a 10-way distinction between relation types. We also provide evidence that the exploitation of the sentences surrounding a citation to a paper can yield higher accuracy than other sentences.

1 Introduction

Identifying the interactions between proteins is one of the most important challenges in modern genomics, with applications throughout cell biology, including expression analysis, signaling, and rational drug design. Most biomedical research and new discoveries are available electronically but only in free text format, so automatic mechanisms are needed to convert text into more structured forms. The goal of this paper is to address this difficult and important task, the extraction of the interactions between proteins from free text. We use graphical models and a neural net that were found to achieve high accuracy in the related task of extracting the relation types might hold between the entities "treatment" and "disease" (Rosario and Hearst, 2004).

Labeling training and test data is time-consuming and subjective. Here we report on results using an existing curated database, the HIV-1 Human Protein Interaction Database[1], to train and test the classification system. The accuracies obtained by the classification models proposed are quite high, confirming the validity of the approach. We also find support for the hypothesis that the sentences surrounding citations are useful for extraction of key information from technical articles (Nakov et al., 2004).

In the remainder of this paper we discuss related work, describe the dataset, and show the results of the algorithm on documents and sentences.

2 Related work

There has been little work in general NLP on trying to identify different relations between entities. Many papers that claim to be doing relationship recognition in actuality address the task of role extraction: (usually two) entities are identified and the relationship is *implied* by the co-occurrence of these entities or by some linguistic expression (Agichtein and Gravano, 2000; Zelenko et al., 2002).

The ACE competition[2] has a relation recognition subtask, but assumes a particular type of relation holds between particular entity types (e.g., if the two entities in question are an EMP and an ORG, then an employment relation holds between them; which type of employment relation depends on the type of entity, e.g., staff person vs partner).

[1] www.ncbi.nlm.nih.gov/RefSeq/HIVInteractions/index.html
[2] http://www.itl.nist.gov/iaui/894.01/tests/ace/

In the BioNLP literature there have recently been a number of attempts to automatically extract protein-protein interactions from PubMed abstracts. Some approaches simply report that a relation exists between two proteins but do not determine which relation holds (Bunescu et al., 2005; Marcotte et al., 2001; Ramani et al., 2005), while most others start with a list of interaction verbs and label only those sentences that contain these trigger words (Blaschke and Valencia, 2002; Blaschke et al., 1999; Rindflesch et al., 1999; Thomas et al., 2000; Sekimizu et al., 1998; Ahmed et al., 2005; Phuong et al., 2003; Pustejovsky et al., 2002). However, as Marcotte et al. (2001) note, "... searches for abstracts containing relevant keywords, such as interact*, poorly discriminate true hits from abstracts using the words in alternate senses and miss abstracts using different language to describe the interactions."

Most of the existing methods also suffer from low recall because they use hand-built specialized templates or patterns (Ono et al., 2001; Corney et al., 2004). Some systems use link grammars in conjunction with trigger verbs instead of templates (Ahmed et al., 2005; Phuong et al., 2003). Every paper evaluates on a different test set, and so it is quite difficult to compare systems.

In this paper, we use state-of-the-art machine learning methods to determine the interaction *types* and to extract the proteins involved. We do not use trigger words, templates, or dictionaries.

3 Data

We use the information from a domain-specific database to gather labeled data for the task of classifying the interactions between proteins in text. The manually-curated HIV-1 Human Protein Interaction Database provides a summary of documented interactions between HIV-1 proteins and host cell proteins, other HIV-1 proteins, or proteins from disease organisms associated with HIV or AIDS. We use this database also because it contains information about the *type* of interactions, as opposed to other protein interaction databases (BIND, MINT, DIP, for example[3]) that list the protein pairs interacting, without specifying the type of interactions.

Interaction	**#Triples**	**Interaction**	**#Triples**
Interacts with	1115	*Complexes with*	45
Activates	778	*Modulates*	43
Stimulates	659	*Enhances*	41
Binds	647	*Stabilizes*	34
Upregulates	316	*Myristoylated by*	34
Imported by	276	*Recruits*	32
Inhibits	194	*Ubiquitinated by*	29
Downregulates	124	*Synergizes with*	28
Regulates	86	*Co-localizes with*	27
Phosphorylates	81	*Suppresses*	24
Degrades	73	*Competes with*	23
Induces	52	*Requires*	22
Inactivates	51		

Table 1: Number of triples for the most common interactions of the HIV-1 database, after removing the distinction in directionality and the triples with more than one interaction.

In this database, the definitions of the interactions depend on the proteins involved and the articles describing the interactions; thus there are several definitions for each interaction type. For the interaction *bind* and the proteins *ANT* and *Vpr*, we find (among others) the definition "*Interaction of HIV-1 Vpr with human adenine nucleotide translocator (ANT) is presumed based on a specific binding interaction between Vpr and rat ANT.*"

The database contains 65 types of interactions and 809 proteins for which there is interaction information, for a total of 2224 pairs of interacting proteins. For each documented protein-protein interaction the database includes information about:

- A pair of proteins (PP),
- The interaction type(s) between them (I), and
- PubMed identification numbers of the journal article(s) describing the interaction(s) (A).

A protein pair PP can have multiple interactions (for example, AIP1 *binds* to HIV-1 p6 and also *is incorporated* into it) for an average of 1.9 interactions per PP and a maximum of 23 interactions for the pair CDK9 and tat p14.

We refer to the combination of a protein pair PP and an article A as a "triple." Our goal is to automatically associate to each triple an interaction

[3]DIP lists only the protein pairs, BIND has only some information about the method used to provide evidence for the interaction, and MIND does have interaction type information but the vast majority of the entries (99.9% of the 47,000 pairs) have been assigned the same type of interaction (*aggregation*). These databases are all manually curated.

type. For the example above, the triple “AIP1 HIV-1-p6 14519844” is assigned the interaction *binds* (14519844 being the PubMed number of the paper providing evidence for this interaction)[4].

Journal articles can contain evidence for multiple interactions: there are 984 journal articles in the database and on average each article is reported to contain evidence for 5.9 triples (with a maximum number of 90 triples).

In some cases the database reports multiple different interactions for a given triple. There are 5369 unique triples in the database and of these 414 (7.7%) have multiple interactions. We exclude these triples from our analysis; however, we do include articles and PPs with multiple interactions. In other words, we tackle cases such as the example above of the pair AIP1, HIV-1-p6 (that can both *bind* and *incorporate*) as long as the evidence for the different interactions is given by two different articles.

Some of the interactions differ only in the directionality (e.g., *regulates* and *regulated by*, *inhibits* and *inhibited by*, etc.); we collapsed these pairs of related interactions into one[5]. Table 1 shows the list of the 25 interactions of the HIV-1 database for which there are more than 10 triples.

For these interactions and for a random subset of the protein pairs PP (around 45% of the total pairs in the database), we downloaded the corresponding full-text papers. From these, we extracted all and only those sentences that contain both proteins from the indicated protein pair. We assigned each of these sentences the corresponding interaction I from the database (“papers”).

Nakov et al. (2004) argue that the sentences surrounding citations to related work, or *citances*, are a useful resource for bioNLP. Building on that work, we use citances as an additional form of evidence to determine protein-protein interaction types. For a given database entry containing PubMed article A, protein pair PP, and interaction type I, we downloaded a subset of the papers that cite A. From these citing papers, we extracted all and only those sentences that mention A explicitly; we further filtered these to include all and only the sentences that contain PP. We labeled each of these sentences with interaction type I (“citances”).

There are often many different names for the same protein. We use LocusLink[6] protein identification numbers and synonym names for each protein, and extract the sentences that contain an exact match for (some synonym of) each protein. By being conservative with protein name matching, and by not doing co-reference analysis, we miss many candidate sentences; however this method is very precise.

On average, for “papers,” we extracted 0.5 sentences per triple (maximum of 79) and 50.6 sentences per interaction (maximum of 119); for “citances” we extracted 0.4 sentences per triple (with a maximum of 105) and 49.2 sentences per interaction (162 maximum). We required a minimum number (40) of sentences for each interaction type for both “papers” and “citances”; the 10 interactions of Table 2 met this requirement. We used these sentences to train and test the models described below[7].

Since all the sentences extracted from one triple are assigned the same interaction, we ensured that sentences from the same triple did not appear in both the testing and the training sets. Roughly 75% of the data were used for training and the rest for testing.

As mentioned above the goal is to automatically associate to each triple an interaction type. The task tackled here is actually slightly more difficult: given some sentences extracted from article A, assign to A an interaction type I and extract the proteins PP involved. In other words, for the purpose of classification, we act as if we do not have information about the proteins that interact. However, given the way the sentence extraction was done, all the sentences extracted from A contain the PP.

[4]To be precise, there are for this PP (as there are often) multiple articles (three in this case) describing the interaction *binds*, thus we have the following three triples to which we associate *binds*: “AIP1 HIV-1-p6 14519844,” “AIP1 HIV-1-p6 14505570” and “AIP1 HIV-1-p6 14505569.”

[5]We collapsed these pairs because the directionality of the interactions was not always reliable in the database. This implies that for some interactions, we are not able to infer the *different* roles of the two proteins; we considered only the pair “prot1 prot2” or “prot2 prot1,” not both. However, our algorithm can detect *which* proteins are involved in the interactions.

[6]LocusLink was recently integrated into Entrez Gene, a unified query environment for genes (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene).

[7]We also looked at larger chunks of text, in particular, we extracted the sentence containing the PP along with the previous and the following sentences, and the three consecutive sentences that contained the PP (the proteins could appear in any of the sentences). However, the results obtained by using these larger chunks were consistently worse.

Interaction	Papers	Citances
Degrades	60	63
Synergizes with	86	101
Stimulates	103	64
Binds	98	324
Inactivates	68	92
Interacts with	62	100
Requires	96	297
Upregulates	119	98
Inhibits	78	84
Suppresses	51	99
Total	821	1322

Table 2: Number of interaction sentences extracted.

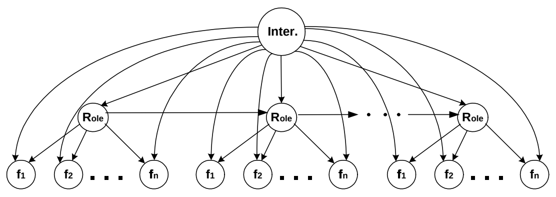


Figure 1: Dynamic graphical model (DM) for protein interaction classification (and role extraction).

A hand-assessment of the individual sentences shows that not every sentence that mentions the target proteins PP actually describes the interaction I (see Section 5.4). Thus the evaluation on the test set is done at the document level (to determine if the algorithm can predict the interaction that a curator would assign to a document as a whole given the protein pair).

Note that we assume here that the papers that provide the evidence for the interactions are given – an assumption not usually true in practice.

4 Models

For assigning interactions, we used two generative graphical models and a discriminative model. Figure 1 shows the generative dynamic model, based on previous work on role and relation extraction (Rosario and Hearst, 2004) where the task was to extract the entities TREATMENT and DISEASE and the relationships between them. The nodes labeled "Role" represent the entities (in this case the choices are PROTEIN and NULL); the children of the role nodes are the words (which act as features), thus there are as many role states as there are words in the sentence; this model consists of a Markov sequence of states where each state generates one or multiple observations. This model makes the additional assumption that there is an interaction present in the sentence (represented by the node "Inter.") that generates the role sequence and the observations. (We assume here that there is a single interaction for each sentence.) The "Role" nodes can be observed or hidden. The results reported here were obtained using only the words as features (i.e., in the dynamic model of Figure 1 there is only one feature node per role) and with the "Role" nodes hidden (i.e., we had no information regarding which proteins were involved). Inference is performed with the junction tree algorithm[8].

We used a second type of graphical model, a simple Naive Bayes, in which the node representing the interaction generates the observable features (all the words in the sentence). We did not include role information in this model.

We defined joint probability distributions over these models, estimated using maximum likelihood on the training set with a simple absolute discounting smoothing method. We performed 10-fold cross validation on the training set and we chose the smoothing parameters for which we obtained the best classification accuracies (averaged over the ten runs) on the training data; the results reported here were obtained using these parameters on the held-out test sets[9].

In addition to these two generative models, we also used a discriminative model, a neural network. We used the Matlab package to train a feed-forward network with conjugate gradient descent. The network has one hidden layer, with a hyperbolic tangent function, and an output layer representing the relationships. A logistic sigmoid function is used in the output layer. The network was trained for several choices of numbers of hidden units; we chose the best-performing networks based on training set error. We then tested these networks on held-out testing data. The features were words, the same as those used for the graphical models.

[8]Using Kevin Murphy's BNT package: http://www.cs.ubc.ca/˜murphyk/Software/BNT/bnt.html.

[9]We did not have enough data to require that the sentences in the training and test sets *of the cross validation procedure* originate from disjoint triples (they do originate from disjoint triple in the final held out data). This may result in a less than optimal choice of the parameters for the aggregate measures described below.

	All		Papers		Citances	
	Mj	**Cf**	**Mj**	**Cf**	**Mj**	**Cf**
DM	60.5	59.7	**57.8**	55.6	53.4	54.5
NB	58.1	61.3	**57.8**	55.6	55.7	54.5
NN	**63.7**	–	44.4	–	**55.8**	–
Key	20.1	–	24.4	–	20.4	–
KeyB	25.8	–	40.0	–	26.1	–
Base.	21.8		11.1		26.1	

Table 3: Accuracies for classification of the 10 protein-protein interactions of Table 2. DM: dynamic model, NB: Naive Bayes, NN: neural network. Baselines: Key: trigger word approach, KeyB: trigger word with backoff, Base: the accuracy of choosing the most frequent interaction.

The task is the following: given a triple consisting of a PP and an article, extract the sentences from the article that contain both proteins. Then, predict for the entire document one of the interactions of Table 2 given the sentences extracted for that triple. This is a 10-way classification problem, which is significantly more complex than much of the related work in which the task is to make the binary prediction (see Section 2).

5 Results

The evaluation was done on a document-by-document basis. During testing, we choose the interaction using the following aggregate measures that use the constraint that all sentences coming from the same triple are assigned the same interaction.

- **Mj**: For each triple, for each sentence of the triple, find the interaction that maximizes the posterior probability of the interaction given the features; then assign to *all* sentences of this triple the most frequent interaction among those predicted for the individual sentences.

- **Cf**: Retain all the conditional probabilities (do not choose an interaction per sentence), then, for each triple, choose the interaction that maximizes the sum over all the triple's sentences.

Table 3 reports the results in terms of classification accuracies averaged across all interactions, for the cases "all" (sentences from "papers" and "citances" together), only "papers" and only "citances". The accuracies are quite high; the dynamic model achieves around 60% for "all," 58% for "papers" and 54% for "citances." The neural net achieves the best results for "all" with around 64% accuracy. From these results we can make the following observations: all models greatly outperform the baselines; the performances of the dynamic model DM, the Naive Bayes NB and the NN are very similar; for "papers" the best results were obtained with the graphical models; for "all" and "citances" the neural net did best. The use of "citances" allowed the gathering of additional data (and therefore a larger training set) that lead to higher accuracies (see "papers" versus "all").

In the confusion matrix in Table 5 we can see the accuracies for the individual interactions for the dynamic model DM, using "all" and "Mj." For three interactions this model achieves perfect accuracy.

5.1 Hiding the protein names

In order to ensure that the algorithm was not overfitting on the protein names, we ran an experiment in which we replaced the protein names in all sentences with the token "PROT_NAME." For example, the sentence: *"Selective CXCR4 antagonism by Tat"* became: *"Selective PROT_NAME2 antagonism by PROT_NAME1."*

Table 5.1 shows the results of running the models on this data. For "papers" and "citances" there is always a decrease in the classification accuracy when we remove the protein names, showing that the protein names do help the classification. The differences in accuracy in the two cases using "citances" are much smaller than the differences using "papers" at least for the graphical models. This suggests that citation sentences may be more robust for some language processing tasks and that the models that use "citances" learn better the linguistic context of the interactions. Note how in this case the graphical models always outperform the neural network.

5.2 Using a "trigger word" approach

As mentioned above, much of the related work in this field makes use of "trigger words" or "interaction words" (see Section 2). In order to (roughly) compare our work and to build a more realistic baseline, we created a list of 70 keywords that are repre-

	Prediction										Acc.
Truth	***D***	***SyW***	***St***	***B***	***Ina***	***IW***	***R***	***Up***	***Inh***	***Su***	**(%)**
Degrades (D)	5	0	0	0	0	0	0	0	0	0	100.0
Synergizes with (SyW)	0	1	0	0	0	1	0	3	3	0	12.5
Stimulates (St)	0	0	4	0	0	0	6	0	1	0	36.4
Binds (B)	0	0	0	18	0	4	1	1	3	0	66.7
Inactivates (Ina)	0	0	0	0	9	0	0	0	0	0	100.0
Interacts with (IW)	0	0	4	3	0	5	1	0	1	2	31.2
Requires (R)	0	0	0	0	0	3	3	0	1	1	37.5
Upregulates (Up)	0	0	0	2	1	0	0	12	2	0	70.6
Inhibits (Inh)	0	0	0	3	0	0	1	1	12	0	70.6
Suppresses (Su)	0	0	0	0	0	0	0	0	0	6	100.0

Table 4: Confusion matrix for the dynamic model DM for "all," "Mj." The overall accuracy is 60.5%. The numbers indicate the number of articles A (each paper has several relevant sentences).

	All			**Papers**			**Citances**		
	Mj	**Cf**	**Diff**	**Mj**	**Cf**	**Diff**	**Mj**	**Cf**	**Diff**
DM	**60.5**	**60.5**	0.7%	44.4	40.0	-25.6%	52.3	**53.4**	-2.0%
NB	59.7	59.7	0.1%	46.7	**51.1**	-11.7%	**53.4**	**53.4**	-3.1%
NN	51.6		-18.9%	44.4		0%	50.0		-10.4%

Table 5: Accuracies for the classification of the 10 protein-protein interactions of Table 2 with the *protein names removed*. Columns marked Diff show the difference in accuracy (in percentages) with respect to the original case of Table 3, averaged over all evaluation methods.

sentative of the 10 interactions. For example, for the interaction *degrade* some of the keywords are "degradation," "degrade," for *inhibit* we have "inhibited," "inhibitor," "inhibitory" and others. We then checked whether a sentence contained such keywords. If it did, we assigned to the sentence the corresponding interaction. If it contained more than one keyword corresponding to multiple interactions consisting of the generic *interact with* plus a more specific one, we assigned the more specific interaction; if the two predicted interactions did not include *interact with* but two more specific interactions, we did not assign an interaction, since we wouldn't know how to choose between them. Similarly, we assigned no interaction if there were more than two predicted interactions or no keywords present in the sentence. The results are shown in the rows labeled "Key" and "KeyB" in Table 3. Case "KeyB" is the "Key" method with back-off: when no interaction was predicted, we assigned to the sentence the most frequent interaction in the training data. As before, we calculated the accuracy when we force all the sentences from one triple to be assign to the most frequent interaction among those predicted for the individual sentences.

KeyB is more accurate than Key and although the KeyB accuracies are higher than the other baselines, they are significantly lower than those obtained with the trained models. The low accuracies of the trigger-word based methods show that the relation classification task is nontrivial, in the sense that not all the sentences contain the most obvious word for the interactions, and suggests that the trigger word approach is insufficient.

5.3 Protein extraction

The dynamic model of Figure 1 has the appealing property of simultaneously performing interaction recognition and protein name tagging (also known as role extraction): the task consists of identifying all the proteins present in the sentence, given a sequence of words. We assessed a slightly different task: the identification of all (and only) the proteins present in the sentence *that are involved in the interaction*.

The F-measure[10] achieved by this model for this task is 0.79 for "all," 0.67 for "papers" and 0.79 for "citances"; again, the model parameters were chosen with cross validation on the training set, and "ci-

[10]The F-measure is a weighted combination of precision and recall. Here, precision and recall are given equal weight, that is, F-measure = $(2 * PRE * REC)/(PRE + REC)$.

tances" had superior performance. Note that we did not use a dictionary: the system learned to recognize the protein names using only the training data. Moreover, our role evaluation is quite strict: every token is assessed and we do not assign partial credit for constituents for which only some of the words are correctly labeled. We also did not use the information that all the sentences extracted from one triple contain the same proteins.

Given these strong results (both F-measure and classification accuracies), we believe that the dynamic model of Figure 1 is a good model for performing both name tagging and interaction classification simultaneously, or either of these task alone.

5.4 Sentence-level evaluation

In addition to assigning interactions to protein pairs, we are interested in sentence-level semantics, that is, in determining the interactions that are actually expressed in the sentence. To test whether the information assigned to the entire document by the HIV-1 database record can be used to infer information at the sentence level, an annotator with biological expertise hand-annotated the sentences from the experiments. The annotator was instructed to assign to each sentence one of the interactions of Table 2, "not interacting," or "other" (if the interaction between the two proteins was not one of Table 2).

Of the 2114 sentences that were hand-labeled, 68.3% of them disagreed with the HIV-1 database label, 28.4% agreed with the database label, and 3.3% were found to contain multiple interactions between the proteins. Among the 68.3% of the sentences for which the labels did not agree, 17.4% had the vague *interact with* relation, 7.4% did not contain any interaction and 43.5% had an interaction different from that specified by the triple[11]. In Table 6 we report the mismatch between the two sets of labels. The total accuracy of 38.9%[12] provides a useful baseline for using a database for the labeling at the sentence level. It may be the case that certain interactions tend to be biologically related and thus tend to co-occur (*upregulate* and *stimulate* or *inactivate* and *inhibit*, for example).

[11]For 28% of the triples, none of the sentences extracted from the target paper were found by the annotator to contain the interaction given by the database. We read four of these papers and found sentences containing that interaction, but our system had failed to extract them.

[12]The accuracy without the vague *interact with* is 49.4%.

	All	**Papers**	**Citan.**
DM	48.9	28.9	47.9
NB	47.1	33.3	53.4
NN	**52.9**	**36.7**	**63.2**
Key	30.5	18.9	38.3
KeyB	46.2	36.3	52.6
Base	36.3	34.4	37.6

Table 7: Classification accuracies when the models are trained and tested on the hand labeled sentences.

We investigated a few of the cases in which the labels were "suspiciously" different, for example a case in which the database interaction was *stimulate* but the annotator found the same proteins to be related by *inhibit* as well. It turned out that the authors of the article assigned *stimulate* found little evidence for this interaction (in favor of *inhibit*), suggesting an error in the database. In another case the database interaction was *require* but the authors of the article, while supporting this, found that under certain conditions (when a protein is too abundant) the interaction changes to one of *inhibit*. Thus we were able to find controversial facts about protein interactions just by looking at the confusion matrix of Table 6.

We trained the models using these hand-labeled sentences in order to determine the interaction expressed *for each sentence* (as opposed to for each document). This is a difficult task; for some sentences it took the annotator several minutes to understand them and decide which interaction applied. Table 7 shows the results on running the classification models on the six interactions for which there were more than 40 examples in the training sets. Again, the sentences from "papers" are especially difficult to classify; the best result for "papers" is 36.7% accuracy versus 63.2% accuracy for "citances." In this case the difference in performance of "papers" and "citances" is larger than for the previous task of document-level relation classification.

6 Conclusions

We tackled an important and difficult task, the classification of different interaction types between proteins in text. A solution to this problem would have an impact on a variety of important challenges in modern biology. We used a protein-interaction

	Annotator											
Database	***D***	***SyW***	***St***	***B***	***Ina***	***R***	***Up***	***Inh***	***Su***	***IW***	***Ot***	***No***
Degrades (D)	44	0	2	5	6	5	2	0	23	9	11	6
Synergizes with (SyW)	0	78	3	14	0	13	8	0	0	26	31	11
Stimulates (St)	0	5	23	12	0	8	7	5	1	26	60	18
Binds (B)	0	6	9	118	0	25	8	10	1	129	77	22
Inactivates (Ina)	0	0	4	25	0	2	4	33	6	14	27	11
Requires (R)	0	5	29	20	0	63	8	54	0	85	80	33
Upregulates (Up)	0	4	24	0	0	0	124	2	0	21	32	4
Inhibits (Inh)	0	8	4	8	2	2	2	43	9	24	37	19
Suppresses (Su)	3	0	0	1	5	0	0	42	34	33	24	4
Interacts with (IW)	0	1	5	28	1	12	6	1	1	49	27	28
Accuracy	93.6	72.9	22.3	51.1	0	48.5	73.4	22.7	45.3	11.8		

Table 6: Confusion matrix comparing the hand-assigned interactions and those extracted from the HIV-1 database. Ot: sentences for which the annotator found an interaction different from those in Table 2. No: sentences for which the annotator found no interaction. The bottom row shows the accuracy of using the database to label the individual sentences.

database to automatically gather labeled data for this task, and implemented graphical models that can simultaneously perform protein name tagging and relation identification, achieving high accuracy on both problems. We also found evidence supporting the hypothesis that citation sentences are a good source of training data, most likely because they provide a concise and precise way of summarizing facts in the bioscience literature.

Acknowledgments. We thank Janice Hamer for her help in labeling examples and other biological insights. This research was supported by a grant from NSF DBI-0317510 and a gift from Genentech.

References

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. *Proc. of DL '00*.

S. Ahmed, D. Chidambaram, H. Davulcu, and C. Baral. 2005. Intex: A syntactic role driven protein-protein interaction extractor for bio-medical text. In *Proceedings ISMB/ACL Biolink 2005*.

C. Blaschke and A. Valencia. 2002. The frame-based module of the suiseki information extraction system. *IEEE Intelligent Systems*, 17(2).

C. Blaschke, M.A. Andrade, C. Ouzounis, and A. Valencia. 1999. Automatic extraction of biological information from scientific text: Protein-protein interactions. *Proc. of ISMB*.

R. Bunescu, R. Ge, R. Kate, E. Marcotte, R. J. Mooney, A. K. Ramani, and Y. W. Wong. 2005. Comparative experiments on learning information extractors for protiens and their interactions. *Artificial Intelligence in Medicine*, 33(2).

D. Corney, B. Buxton, W. Langdon, and D. Jones. 2004. Biorat: extracting biological information from full-length papers. *Bioinformatics*, 20(17).

E. Marcotte, I. Xenarios, and D. Eisenberg. 2001. Mining literature for protein-protein interactions. *Bioinformatics*, 17(4).

P. Nakov, A. Schwartz, and M. Hearst. 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *Proceedings of the SIGIR'04 workshop on Search and Discovery in Bioinformatics*.

T. Ono, H. Hishigaki, A. Tanigami, and T. Takagi. 2001. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(1).

T. Phuong, D. Lee, and K-H. Lee. 2003. Learning rules to extract protein interactions from biomedical text. In *PAKDD*.

J. Pustejovsky, J. Castano, and J. Zhang. 2002. Robust relational parsing over biomedical literature: Extracting inhibit relations. *Proc. of Pac Symp Biocomputing*.

C. Ramani, E. Marcotte, R. Bunescu, and R. Mooney. 2005. Using biomedical literature mining to consolidate the set of known human protein-protein interactions. In *Proceedings ISMB/ACL Biolink 2005*.

T. Rindflesch, L. Hunter, and L. Aronson. 1999. Mining molecular binding terminology from biomedical text. *Proceedings of the AMIA Symposium*.

Barbara Rosario and Marti A. Hearst. 2004. Classifying semantic relations in bioscience texts. In *Proc. of ACL 2004*.

T. Sekimizu, H.S. Park, and J. Tsujii. 1998. Identifying the interaction between genes and gene products based on frequently seen verbs in medline abstracts. *Gen. Informat.*, 9.

J. Thomas, D. Milward, C. Ouzounis, and S. Pulman. 2000. Automatic extraction of protein interactions from scientific abstracts. *Proceedings of the Pac Symp Biocomput*.

D. Zelenko, C. Aone, and A. Richardella. 2002. Kernel methods for relation extraction. *Proceedings of EMNLP 2002*.

BLANC[1]: Learning Evaluation Metrics for MT

Lucian Vlad Lita and **Monica Rogati** and **Alon Lavie**
Carnegie Mellon University
{llita,mrogati,alavie}@cs.cmu.edu

Abstract

We introduce BLANC, a family of dynamic, trainable evaluation metrics for machine translation. Flexible, parametrized models can be learned from past data and automatically optimized to correlate well with human judgments for different criteria (e.g. adequacy, fluency) using different correlation measures. Towards this end, we discuss ACS (all common skip-ngrams), a practical algorithm with trainable parameters that estimates reference-candidate translation overlap by computing a weighted sum of all common skip-ngrams in polynomial time. We show that the BLEU and ROUGE metric families are special cases of BLANC, and we compare correlations with human judgments across these three metric families. We analyze the algorithmic complexity of ACS and argue that it is more powerful in modeling both local meaning and sentence-level structure, while offering the same practicality as the established algorithms it generalizes.

1 Introduction

Although recent MT evaluation methods show promising correlations to human judgments in terms of adequacy and fluency, there is still considerable room for improvement (Culy and Riehemann, 2003). Most of these studies have been performed at a system level and have not investigated metric robustness at a lower granularity. Moreover, even though the emphasis on adequacy vs. fluency is application-dependent, automatic evaluation metrics do not distinguish between the need to optimize correlation with regard to one or the other.

Machine translation automatic evaluation metrics face two important challenges: the lack of powerful features to capture both sentence level structure and local meaning, and the difficulty of designing good functions for combining these features into meaningful quality estimation algorithms.

In this paper, we introduce BLANC[1], an automatic MT evaluation metric family that is a generalization of popular and successful metric families currently used in the MT community (BLEU, ROUGE, F-measure etc.). We describe an efficient, polynomial-time algorithm for BLANC, and show how it can be optimized to target adequacy, fluency or any other criterion. We compare our metric's performance with traditional and recent automatic evaluation metrics. We also describe the parameter conditions under which BLANC can emulate them.

Throughout the remainder of this paper, we distinguish between two components of automatic MT evaluation: the *statistics* computed on candidate and reference translations and the *function* used in defining evaluation metrics and generating translation scores. Commonly used statistics include bag-of-words overlap, edit distance, longest common subsequence, ngram overlap, and skip-bigram overlap. Preferred functions are various combinations of precision and recall (Soricut and Brill, 2004), including

[1]Since existing evaluation metrics (e.g. BLEU, ROUGE) are special cases of our metric family, it is only natural to name it Broad Learning and Adaptation for Numeric Criteria (BLANC) – white light contains light of all frequencies

weighted precision and F-measures (Van-Rijsbergen, 1979).

BLANC implements a practical algorithm with learnable parameters for automatic MT evaluation which estimates the reference-candidate translation overlap by computing a weighted sum of common subsequences (also known as skip-ngrams). Common skip-ngrams are sequences of words in their sentence order that are found both in the reference and candidate translations. By generalizing and separating the overlap statistics from the function used to combine them, and by identifying the latter as a learnable component, BLANC subsumes the ngram based evaluation metrics as special cases and can better reflect the need of end applications for adequacy/fluency tradeoffs .

1.1 Related Work

Initial work in evaluating translation quality focused on edit distance-based metrics (Su et al., 1992; Akiba et al., 2001). In the MT context, edit distance (Levenshtein, 1965) represents the amount of word insertions, deletions and substitutions necessary to transform a candidate translation into a reference translation. Another evaluation metric based on edit distance is the *Word Error Rate* (Niessen et al., 2000) which computes the normalized edit distance. BLEU is a weighted precision evaluation metric introduced by IBM (Papineni et al., 2001). BLEU and its extensions/variants (e.g. NIST (Doddington, 2002)) have become de-facto standards in the MT community and are consistently being used for system optimization and tuning. These methods rely on local features and do not explicitly capture sentence-level features, although implicitly longer n-gram matches are rewarded in BLEU. The General Text Matcher (GTM) (Turian et al., 2003) is another MT evaluation method that rewards longer ngrams instead of assigning them equal weight.

(Lin and Och, 2004) recently proposed a set of metrics (ROUGE) for MT evaluation. ROUGE-L is a longest common subsequence (LCS) based automatic evaluation metric for MT. The intuition behind it is that long common subsequences reflect a large overlap between a candidate translation and a reference translation. ROUGE-W is also based on LCS, but assigns higher weights to sequences that have fewer gaps. However, these metrics still do not distinguish among translations with the same LCS but different number of shorter sized subsequences, also indicative of overlap. ROUGE-S attempts to correct this problem by combining the precision/recall of skip-*bigrams* of the reference and candidate translations. However, by using skip-ngrams with n¿=2, we might be able to capture more information encoded in the higher level sentence structure. With BLANC, we propose a way to exploit local contiguity in a manner similar to BLEU and also higher level structure similar to ROUGE type metrics.

2 Approach

We have designed an algorithm that can perform a full overlap search over variable-size, non-contiguous word sequences (skip-ngrams) efficiently. At first glance, in order to perform this search, one has to first exhaustively generate all skip-ngrams in the candidate and reference segments and then assess the overlap. This approach is highly prohibitive since the number of possible sequences is exponential in the number of words in the sentence. Our algorithm – ACS (all common skip-ngrams) – directly constructs the set of overlapping skip-ngrams through incremental composition of word-level matches. With ACS, we can reduce computation complexity to a fifth degree polynomial in the number of words.

Through the ACS algorithm, BLANC is not limited only to counting skip-ngram overlap: the contribution of different skip-ngrams to the overall score is based on a set of features. ACS computes the overlap between two segments of text and also allows local and global features to be computed during the overlap search. These local and global features are subsequently used to train evaluation models within the BLANC family. We introduce below several simple skip-ngram-based features and show that special-case parameter settings for these features emulate the computation of existing ngram-based metrics. In order to define the relative significance of a particular skip-ngram found by the ACS algorithm, we employ an exponential model for feature integration.

2.1 Weighted Skip-Ngrams

We define *skip-ngrams* as sequences of n words taken in sentence order allowing for arbitrary gaps. In algorithms literature skip-ngrams are equivalent to *subsequences*. As special cases, skip-ngrams with n=2 are referred to as skip-bigrams and skip-ngrams with no gaps between the words are simply $ngrams$. A sentence S of size $|S|$ has $C(|S|, n) = \frac{|S|!}{(|S|-n)!n!}$ skip-ngrams.

For example, the sentence "*To be or not to be*" has $C(6, 2) = 15$ corresponding skip-bigrams including "*be or*", "*to to*", and three occurrences of "*to be*". It also has $C(6, 4) = 15$ corresponding skip-4grams ($n = 4$) including "to be to be" and "to or not to".

Consider the following sample reference and candidate translations:

R_0:	machine translated text is evaluated automatically
K_1:	machine translated stories are chosen automatically
K_2:	machine and human together can forge a friendship that cannot be translated into words automatically
K_3:	machine code is being translated automatically

The skip-ngram "*machine translated automatically*" appears in both the reference R_0 and all candidate translations. Arguably, a skip-bigram that contains few gaps is likely to capture local structure or meaning. At the same time, skip-ngrams spread across a sentence are also very useful since they may capture part of the high level sentence structure.

We define a **weighting** feature function for skip-ngrams that estimates how likely they are to capture local meaning and sentence structure. The weighting function φ for a skip-ngram w_1 ..w_n is defined as:

$$\varphi(w_1..w_n) = e^{-\alpha \cdot G(w_1..w_n)} \tag{1}$$

where $\alpha \geq 0$ is a decay parameter and G($w_1..w_n$) measures the overall gap of the skip-ngram $w_1..w_n$ in a specific sentence. This overall skip-ngram weight can be decomposed into the weights of its constituent skip-bigrams:

$$\begin{aligned} \varphi(w_1..w_n) &= e^{-\alpha \cdot G(w_1,..,w_n)} && (2) \\ &= e^{-\alpha \cdot \sum_{i=1}^{n-1} G(w_i, w_{i+1})} \\ &= \prod_{i=1}^{n-1} \varphi(w_i\ w_{i+1}) && (3) \end{aligned}$$

In equation 3, $\varphi(w_i\ w_{i+1})$ is the number of words between w_i and w_{i+1} in the sentence. In the example above, the skip-ngram "machine translated automatically" has weight $e^{-3\alpha}$ for sentence K_1 and weight $e^{-12\alpha} = 1$ for sentence K_2.

In our initial experiments the gap G has been expressed as a linear function, but different families of functions can be explored and their corresponding parameters learned. The parameter α dictates the behavior of the weighting function. When $\alpha = 0$ φ equals $e^0 = 1$, rendering gap sizes irrelevant. In this case, skip-ngrams are given the same weight as contiguous ngrams. When α is very large, φ approaches 0 if there are any gaps in the skip-ngram and is 1 if there are no gaps. This setting has the effect of considering only contiguous ngrams and discarding all skip-ngrams with gaps.

In the above example, although the skip-ngram "machine translated automatically" has the same cumulative gap in both in K_1 and K_3, the occurrence in K_1 has is a gap distribution that more closely reflects that of the reference skip-ngram in R_0. To model gap distribution differences between two occurrences of a skip-ngram, we define a piece-wise distance function δ_{XY} between two sentences x and y. For two successive words in the skip-ngram, the distance function is defined as:

$$\delta_{XY}(w_1 w_2) = e^{-\beta \cdot |G_X(w_1,w_2) - G_Y(w_1,w_2)|} \tag{4}$$

where $\beta \geq 0$ is a decay parameter. Intuitively, the β parameter is used to reward better aligned skip-ngrams. Similar to the φ function, the overall δ_{XY} distance between two occurrences of a skip-ngram with $n > 1$ is:

$$\delta_{XY}(w_1..w_n) = \prod_{i=1}^{n-1} \delta_{XY}(w_i w_{i+1}) \tag{5}$$

Note that equation 5 takes into account pairs of skip-ngrams skip in different places by summing over piecewise differences. Finally, using an exponential model, we assign an overall score to the matched skip-ngram. The skip-ngram scoring function S_{xy} allows independent features to be incorporated into the overall score:

$$\begin{aligned} S_{xy}(w_i..w_k) &= \varphi(w_i..w_k) \cdot \delta_{xy}(w_i..w_k) \\ &\cdot e^{\lambda_1 f_1(w_i..w_k)} \cdot ... \cdot e^{\lambda_h f_h(w_i..w_k)} \end{aligned} \tag{6}$$

where features $f_1..f_h$ can be functions based on the syntax, semantics, lexical or morphological aspects of the skip-ngram. Note that different models for combining skip-ngram features can be used in conjunction with ACS.

2.2 Multiple References

In BLANC we incorporate multiple references in a manner similar to the ROUGE metric family. We compute the precision and recall of each size skip-ngrams for individual references. Based on these we combine the maximum precision and maximum recall of the candidate translation obtained using all reference translations and use them to compute an aggregate F-measure.

The F-measure parameter β_F is modeled by BLANC. In our experiments we optimized β_F individually for fluency and adequacy.

2.3 The ACS Algorithm

We present a practical algorithm for extracting *All Common Skip-ngrams* (ACS) of any size that appear in the candidate and reference translations. For clarity purposes, we present the ACS algorithm as it relates to the MT problem: find all common skip-ngrams (ACS) of any size in two sentences X and Y:

$$wSKIP \leftarrow Acs(\delta, \varphi, X, Y) \quad (7)$$

$$= \{wSKIP_1..wSKIP_{min(|X|,|Y|)}\} \quad (8)$$

where $wSkip_n$ is the set of all skip-ngrams of size n and is defined as:

$$wSKIP_n = \{\text{“}w_1..w_n\text{”} \mid w_i \in X, w_i \in Y, \forall i \in [1..n] \text{ and } w_i \prec w_j, \forall i < j \in [1..n]\}$$

Given two sentences X and Y we observe a *match* (w, x, y) if word w is found in sentence X at index x and in sentence Y at index y:

$$(w, x, y) \equiv \{0 \leq x \leq |X|, 0 \leq y \leq |Y|, w \in V, \textit{and } X[x] = Y[y] = w\} \quad (9)$$

where V is the vocabulary with a finite set of words.

In the following subsections, we present the following steps in the ACS algorithm:

1. *identify all matches* – find matches and generate corresponding nodes in the dependency graph
2. *generate dependencies* – construct edges according to pairwise match dependencies
3. *propagate common subsequences* – count all common skip-ngrams using corresponding weights and distances

In the following sections we use the following example to illustrate the intermediate steps of ACS.

X. “to be or not to be”
Y. “to exist or not be”

2.3.1 Step 1: Identify All Matches

In this step we identify all word matches (w, x, y) in sentences X and Y. Using the example above, the intermediate inputs and outputs of this step are:

Input:	X.	“to be or not to be”
	Y.	“to exist or not be”
Output:	(*to*,1,1); (*to*,5,1); (*or*,3,3); (*be*,2,5); …	

For each match we create a corresponding node N in a dependency graph. With each node we associate the actual word matched and its corresponding index positions in both sentences.

2.3.2 Step 2: Generate Dependencies

A dependency $N_1 \rightarrow N_2$ occurs when the two corresponding matches (w_1, x_1, y_1) and (w_2, x_2, y_2) can form a valid common skip-bigram: i.e. when $x_1 < x_2$ and $y_1 < y_2$. Note that the matches can cover identical words, but their indices cannot be the same ($x_1 \neq x_2$ and $y_1 \neq y_2$) since a skip-bigram requires two different word matches.

In order to facilitate the generation of all common subsequences, the graph is populated with the appropriate dependency edges:

```
for each node N in DAG
  for each node M$\neq$N in DAG
    if N(x)$\leq$M(x) and N(y)$\leq$M(y)
      create edge E: N$\rightarrow$M
        compute $\delta_{XY}$(E)
        compute $\varphi$(E)
```

This step incorporates the concepts of skip-ngram weight and distance into the graph. With each edge $E : N_1 \rightarrow N_2$ we associate step-wise weight and distance information for the corresponding skip-bigram formed by matches (w_1, x_1, y_1) and (w_2, x_2, y_2).

Note that rather than counting all skip-ngrams, which would be exponential in the worst case scenario, we only construct a structure of match dependencies (i.e. skip-bigrams). As in dynamic programming, in order to avoid exponential complexity, we compute individual skip-ngram scores only once.

2.3.3 Step 3: Propagate Common Subsequences

In this last step, the ACS algorithm *counts* all common skip-ngrams using corresponding weights and distances. In the general case, this step is equivalent measuring the overlap of the two sentences X and Y. As a special case, if no features are used, the

ACS algorithm is equivalent to counting the number of common skip-ngrams regardless of gap sizes.

```
// depth first search (DFS)
for each node N in DAG
    compute node N's depth

// initialize skip-ngram counts
for each node N in DAG
    v_N[1] ← 1
    for i=2 to LCS(X,Y)
        v_N[i] = 0

// compute ngram counts
for d=1 to MAXDEPTH
    for each node N of depth d in DAG
        for each edge E: N→M
            for i=2 to d
                v_M[i] += $S_{xy}$($\delta$(E), $\varphi$(E), v_N[i-1])
```

After algorithm ACS is run, the number of skip-ngrams (weighted skip-ngram score) of size k is simply the sum of the number of skip-ngrams of size k ending in each node N's corresponding match:

$$wSKIP_k = \sum_{N_i \in DAG} v_{N_i}[k] \tag{10}$$

2.3.4 ACS Complexity and Feasibility

In the worst case scenario, both sentences X and Y are composed of exactly the same repeated word: X = "*w w w w* .. " and Y = "*w w w w* ..". We let $m = |X|$ and $n = |Y|$. In this case, the number of matches is $M = n \cdot m$. Therefore, Step 1 has worst case time and space complexity of $O(m \cdot n)$. However, empirical data suggest that there are far fewer matches than in the worst-case scenario and the actual space requirements are drastically reduced. Even in the worst-case scenario, if we assume the average sentences is fewer than 100 words, the number of nodes in the DAG would only be $10,000$. Step 2 of the algorithm consists of creating edges in the dependency graph. In the worst case scenario, the number of directed edges is $O(M^2)$ and furthermore if the sentences are uniformly composed of the same repeated word as seen above, the worst-case time and space complexity is $m(m+1)/2 \cdot n(n+1)/2 = O(m^2n^2)$. In Step 3 of the algorithm, the DFS complexity for computing of node depths is $O(M)$ and the complexity of $LCS(X, Y)$ is $O(m \cdot n)$. The dominant step is the propagation of common subsequences (skip-ngram counts). Let l be the size of the LCS. The upper bound on the size of the longest common subsequence is $min(|X|, |Y|) = min(m, n)$. In the worst case scenario, for each node we propagate l count values (the size of vector v) to all other nodes in the DAG. Therefore, the time complexity for Step 3 is $O(M^2 \cdot l) = O(m^2n^2l)$ (fifth degree polynomial).

3 BLANC as a Generalization of BLEU and ROUGE

Due to its parametric nature, the All Common Subsequences algorithm can emulate the ngram computation of several popular MT evaluation metrics. The weighting function φ allows skip-ngrams with different gap sizes to be assigned different weights. Parameter α controls the shape of the weighting function.

In one extreme scenario, if we allow α to take very large values, the net effect is that all contiguous ngrams of any size will have corresponding weights of $e^0 = 1$ while all other skip-ngrams will have weights that are zero. In this case, the distance function will only apply to contiguous ngrams which have the same size and no gaps. Therefore, the distance function will also be 1. The overall result is that the ACS algorithm collects contiguous common ngram counts for all ngram sizes. This is equivalent to computing the ngram overlap between two sentences, which is equivalent to the ngram computation performed BLEU metric. In addition to computing ngram overlap, BLEU incorporates a thresholding (clipping) on ngram counts based on reference translations, as well as a brevity penalty which makes sure the machine-produced translations are not too short. In BLANC, this is replaced by standard F-measure, which research (Turian et al., 2003) has shown it can be used successfully in MT evaluation.

Another scenario consists of setting the α and β parameters to 0. In this case, all skip-ngrams are assigned the same weight value of 1 and skip-ngram matches are also assigned the same distance value of 1 regardless of gap sizes and differences in gap sizes. This renders all skip-ngrams equivalent and the ACS algorithm is reduced to counting the *skip-ngram overlap* between two sentences. Using these counts, precision and recall-based metrics such as the F-measure can be computed. If we let the α and β parameters to be zero, disregard redundant matches, and compute

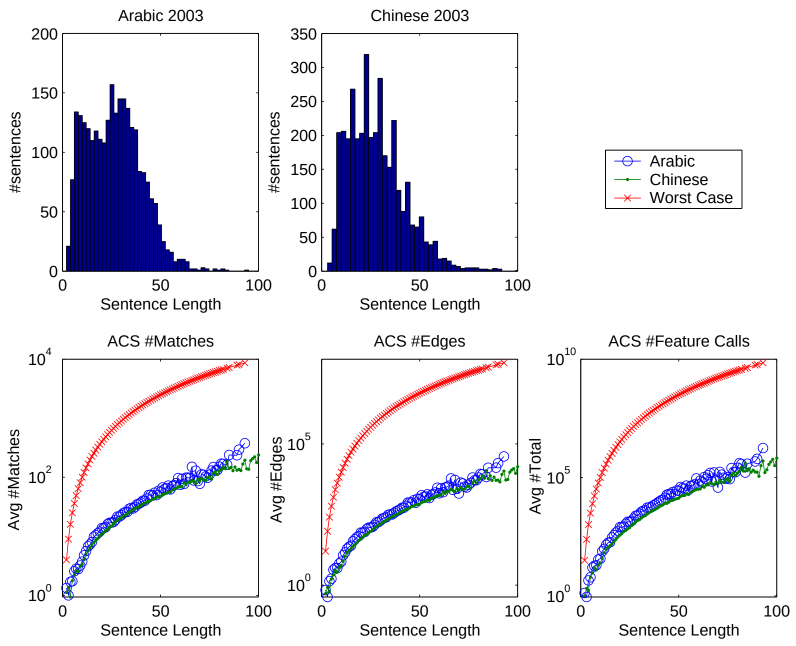


Figure 1: Empirical and theoretical behavior of ACS on 2003 machine translation evaluation data (semilog scale).

the ACS only for skip-ngrams of size 2, the ACS algorithm is equivalent to the ROUGE-S metric (Lin and Och, 2004). This case represents a specific parameter setting in the ACS skip-ngram computation.

The longest common subsequence statistic has also been successfully used for automatic machine translation evaluation in the ROUGE-L (Lin and Och, 2004) algorithm. In BLANC, if we set both α and β parameters to zero, the net result is a set of skip-bigram (common subsequence) overlap counts for all skip-bigram sizes. Although dynamic programming or suffix trees can be used to compute the LCS much faster, under this parameter setting the ACS algorithm can also produce the longest common subsequence:

$$LCS(X,Y) \leftarrow \underset{k}{\text{argmax}}\ ACS(wSKIP_k) > 0$$

where $Acs(wSKIP_k)$ is the number of common skip-ngrams (common subsequences) produced by the ACS algorithm.

ROUGE-W (Lin and Och, 2004) relies on a weighted version of the longest common subsequence, under which longer contiguous subsequences are assigned a higher weight than subsequences that incorporate gaps. ROUGE-W uses the polynomial function x^a in the weighted LCS computation. This setting can also be simulated by BLANC by adjusting the parameters α to reward tighter skip-ngrams and β to assign a very high score to similar size gaps. Intuitively, α is used to reward skip-ngrams that have smaller gaps, while β is used to reward better aligned skip-ngram overlap.

4 Scalability & Data Exploration

In Figure 1 we show theoretical and empirical practical behavior for the ACS algorithm on the 2003 TIDES machine translation evaluation data for Arabic and Chinese. Sentence length distribution is somewhat similar for the two languages – only a very small amount of text segments have more than 50 tokens. We show the ACS graph size in the worst case scenario, and the empirical average number of matches for both languages as a function of sentence length. We also show (on a log scale) the upper bound on time/space complexity in terms of total number of feature computations. Even though the worst-case scenario is tractable (polynomial), the empirical amount of computation is considerably smaller in the form of polynomials of lower degree. In Figure 1, sentence length is the average between reference and candidate lengths.

Finally, we also show the total number of feature computations involved in performing a full overlap search and computing a numeric score for the reference-candidate translation pair. We have experimented with the ACS algorithm using a worst-case scenario where all words are exactly the same for a fifty words reference translation and candidate translation. In practice when considering real sentences the number of matches is very small. In this setting, the algorithm takes less than two seconds on a low-end desktop system when working on the worst case scenario, and less then a second for all candidate-reference pairs in the TIDES 2003 dataset. This result renders the ACS algorithm very practical for automatic MT evaluation.

5 Experiments & Results

In the dynamic metric BLANC, we have implemented the ACS algorithm using several parameters including the aggregate gap size α, the displacement feature β, a parameter for regulating skip-ngram size contribution, and the F-measure β_F parameter.

Until recently, most experiments that evaluate automatic metrics correlation to human judgments have been performed at a system level. In such experiments, human judgments are aggregated across sentences for each MT system and compared to aggregate scores for automatic metrics. While high scoring metrics in this setting are useful for understanding relative system performance, not all of them are robust enough for evaluating the quality of machine translation output at a lower granularity. Sentence-level translation quality estimation is very useful when MT is used as a component in a pipeline of text-processing applications (e.g. question answering). The fact that current automatic MT evaluation metrics including BLANC do not correlate well with human judgments at the sentence level, does not mean we should ignore this need and focus only on system level evaluation. On the contrary, further research is required to improve these metrics. Due to its trainable nature, and by allowing additional features to be incorporated into its model, BLANC has the potential to address this issue.

For comparison purposes with previous literature, we have also performed experiments at system level for Arabic. The datasets used consist of the MT translation outputs from all systems available through the Tides 2003 evaluation (663 sentences) for training and Tides 2004 evaluation (1353 sentences) for testing.

We compare (Table 1) the performance of BLANC on Arabic translation output with the performance of more established evaluation metrics: BLEU and NIST, and also with more recent metrics: ROUGE-L and ROUGE-S (using an unlimited size skip window), which have been shown to correlate well with human judgments at system level – as confirmed by our results. We have performed experiments in which case information is preserved as well as experiments that ignore case information. Since the results are very similar, we only show here experiments under the former condition. In order to maintain consistency, when using any metric we apply the same preprocessing provided by the MTEval script. When computing the correlation between metrics and human judgments, we only keep strictly positive scores. While this is not fully equivalent to BLEU smoothing, it partially mitigates the same problem of zero count ngrams for short sentences. In future work we plan to implement smoothing for all metrics, including BLANC.

We train BLANC separately for adequacy and fluency, as well as for system level and segment level correlation with human judgments. The BLANC parameters are currently trained using a simple hill-climbing procedure and using several starting points in order to decrease the chance of reaching a local maximum.

BLANC proves to be robust across criteria and granularity levels. As expected, different parameter values of BLANC optimize different criteria (e.g. adequacy and fluency). We have observed that training BLANC for adequacy results in more bias towards recall (β_F=3) compared to training it for fluency (β_F=2). This confirms our intuition that a dynamic, parametric metric is justified for automatic evaluation.

6 Conclusions & Future Work

In previous sections we have defined simple distance functions. More complex functions can also be incorporated in ACS. Skip-ngrams in the candidate sentence might be rewarded if they contain fewer gaps in the candidate sentence and penalized if they contain more. Different distance functions could also be used in ACS, including functions based on surface-form features and part-of-speech features.

Most of the established MT evaluation methods are

Tides 2003 Arabic

	System Level		Segment Level	
Method	Adequacy	Fluency	Adequacy	Fluency
BLEU	0.950	0.934	0.382	0.286
NIST	0.962	0.939	0.439	0.304
ROUGE-L	0.974	0.926	0.440	0.328
ROUGE-S	0.949	0.935	0.360	0.328
BLANC	**0.988**	**0.979**	**0.492**	**0.391**

Tides 2004 Arabic

	System Level		Segment Level	
Method	Adequacy	Fluency	Adequacy	Fluency
BLEU	0.978	0.994	0.446	0.337
NIST	**0.987**	0.952	0.529	0.358
ROUGE-L	0.981	0.985	0.538	0.412
ROUGE-S	0.937	0.980	0.367	0.408
BLANC	0.982	**0.994**	**0.565**	**0.438**

Table 1: Pearson correlation of several metrics with human judgments at system level and segment level for fluency and adequacy.

static functions according to which automatic evaluation scores are computed. In this paper, we have laid the foundation for a more flexible, parametric approach that can be trained using existing MT data and that can be optimized for highest agreement with human assessors, for different criteria.

We have introduced ACS, a practical algorithm with learnable parameters for automatic MT evaluation and showed that ngram computation of popular evaluation methods can be emulated through different parameters by ACS. We have computed time and space bounds for the ACS algorithm and argued that while it is more powerful in modeling local and sentence structure, it offers the same practicality as established algorithms.

In our experiments, we trained and tested BLANC on data from consecutive years, and therefore tailored the metric for two different operating points in MT system performance. In this paper we show that BLANC correlates well with human performance when trained on previous year data for both sentence and system level.

In the future, we plan to investigate the stability and performance of BLANC and also apply it to automatic summarization evaluation. We plan to optimize the BLANC parameters for different criteria in addition to incorporating syntactic and semantic features (e.g. ngrams, word classes, part-of-speech).

In previous sections we have defined simple distance functions. More complex functions can also be incorporated in ACS. Skip-ngrams in the candidate sentence might be rewarded if they contain fewer gaps in the candidate sentence and penalized if they contain more. Different distance functions could also be used in ACS, including functions based on surface-form features and part-of-speech features.

Looking beyond the BLANC metric, this paper makes the case for the need to shift to trained, dynamic evaluation metrics which can adapt to individual optimization criteria and correlation functions.

We plan to make available an implementation of BLANC at *http://www.cs.cmu.edu/ llita/blanc*.

References

Y. Akiba, K. Iamamurfa, and E. Sumita. 2001. Using multiple edit distances to automatically rank machine translation output. *MT Summit VIII*.

C. Culy and S.Z. Riehemann. 2003. The limits of n-gram translation evaluation metrics. *Machine Translation Summit IX*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *Human Language Technology Conference (HLT)*.

V.I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*.

C.Y. Lin and F.J. Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip bigram statistics. *ACL*.

S. Niessen, F.J. Och, G. Leusch, and H. Ney. 2000. An evaluation tool for machine translation: Fast evaluation for mt research. *LREC*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. *IBM Research Report*.

R. Soricut and E. Brill. 2004. A unified framework for automatic evaluation using n-gram co-occurence statistics. *ACL*.

K.Y. Su, M.W. Wu, and J.S. Chang. 1992. A new quantitative quality measure for machine translation systems. *COLING*.

J.P. Turian, L. Shen, and I.D. Melamed. 2003. Evaluation of machine translation and its evaluation. *MT Summit IX*.

C.J. Van-Rijsbergen. 1979. Information retrieval.

Composition of Conditional Random Fields for Transfer Learning

Charles Sutton and Andrew McCallum
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{sutton,mccallum}@cs.umass.edu

Abstract

Many learning tasks have subtasks for which much training data exists. Therefore, we want to *transfer* learning from the old, general-purpose subtask to a more specific new task, for which there is often less data. While work in transfer learning often considers how the old task should affect learning on the new task, in this paper we show that it helps to take into account how the new task affects the old. Specifically, we perform joint decoding of separately-trained sequence models, preserving uncertainty between the tasks and allowing information from the new task to affect predictions on the old task. On two standard text data sets, we show that joint decoding outperforms cascaded decoding.

1 Introduction

Many tasks in natural language processing are solved by chaining errorful subtasks. Within information extraction, for example, part-of-speech tagging and shallow parsing are often performed before the main extraction task. Commonly these subtasks have their own standard sets of labeled training data: for example, many large data sets exist for learning to extract person names from newswire text; whereas the available training data for new applications, such as extracting appointment information from email, tends to be much smaller. Thus, we need to transfer regularities learned from a well-studied subtask, such as finding person names in newswire text, to a new, related task, such as finding names of speakers in email seminar announcements.

In previous NLP systems, transfer is often accomplished by training a model for the subtask, and using its prediction as a feature for the new task. For example, recent CoNLL shared tasks (Tjong Kim Sang & De Meulder, 2003; Carreras & Marquez, 2004), which are standard data sets for such common NLP tasks as clause identification and named-entity recognition, include predictions from a part-of-phrase tagger and a shallow parser as features. But including only the single most likely subtask prediction fails to exploit useful dependencies between the tasks. First, if the subtask prediction is wrong, the model for the new task may not be able to recover. Often, errors propagate upward through the chain of tasks, causing errors in the final output. This problem can be ameliorated by preserving uncertainty in the subtask predictions, because even if the best subtask prediction is wrong, the distribution over predictions can still be somewhat accurate.

Second, information from the main task can inform the subtask. This is especially important for learning transfer, because the new domain often has different characteristics than the old domain, which is often a standard benchmark data set. For example, named-entity recognizers are usually trained on newswire text, which is more structured and grammatical than email, so we expect an off-the-shelf named-entity recognizer to perform somewhat worse on email. An email task, however, often has domain-specific features, such as PREVIOUS WORD IS *Speaker:*), which were unavailable or uninformative to the subtask on the old training set, but are very informative to the subtask in the new domain. While previous work in transfer learning has considered how the old task can help the new task, in this paper we show how the new task can help itself by improving predictions on the old.

In this paper we address the issue of transfer by training a cascade of models independently on the various training sets, but at test time combining them into a single model in which decoding is performed jointly. For the individual models, we use linear-chain conditional random fields (CRFs), because the great freedom that they allow in feature engineering facilitates the learning of richer interactions between the subtasks. We train a linear chain CRF on each subtask, using the prediction of the previous subtask as a feature. At test time, we combine the learned weights from the original CRFs into a single grid-shaped factorial CRF, which makes predictions for all the tasks at once. Viterbi decoding in this combined model implicitly considers all possible predictions for the subtask when making decisions in the main task.

We evaluate joint decoding for learning transfer on a standard email data set and a standard entity recognition task. On the email data set, we show a significant gain in performance, including new state-of-the-art results. Of particular interest for transfer learning, we also show that using joint decoding, we achieve equivalent results to cascaded decoding with 25% less training data.

2 Linear-chain CRFs

Conditional random fields (CRFs) (Lafferty et al., 2001) are undirected graphical models that are conditionally trained. In this section, we describe CRFs for the linear-chain case. Linear-chain CRFs can be roughly understood as conditionally-trained finite state machines. A linear-chain CRF defines a distribution over state sequences $\mathbf{s} = \{s_1, s_2, \ldots, s_T\}$ given an input sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ by making a first-order Markov assumption on states. These Markov assumptions imply that the distribution over sequences factorizes in terms of pairwise functions $\Phi_t(s_{t-1}, s_t, \mathbf{x})$ as:

$$p(\mathbf{s}|\mathbf{x}) = \frac{\prod_t \Phi_t(s_{t-1}, s_t, \mathbf{x})}{Z(\mathbf{x})}, \tag{1}$$

The partition function $Z(\mathbf{x})$ is defined to ensure that the distribution is normalized:

$$Z(\mathbf{x}) = \sum_{\mathbf{s}'} \prod_t \Phi_t(s'_{t-1}, s'_t, \mathbf{x}). \tag{2}$$

The *potential functions* $\Phi_t(s_{t-1}, s_t, \mathbf{x})$ can be interpreted as the cost of making a transition from state s_{t-1} to state s_t at time t, similar to a transition probability in an HMM.

Computing the partition function $Z(\mathbf{x})$ requires summing over all of the exponentially many possible state sequences $\mathbf{s}'$. By exploiting Markov assumptions, however, $Z(\mathbf{x})$ (as well as the node marginals $p(s_t|\mathbf{x})$ and the Viterbi labeling) can be calculated efficiently by variants of the standard dynamic programming algorithms used for HMMs.

We assume the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that

$$\Phi(s_{t-1}, s_t, \mathbf{x}) = \exp\left(\sum_k \lambda_k f_k(s_{t-1}, s_t, \mathbf{x}, t)\right). \tag{3}$$

The model parameters are a set of real weights $\Lambda = \{\lambda_k\}$, one for each feature.

Feature functions can be arbitrary. For example, one feature function could be a binary test $f_k(s_{t-1}, s_t, \mathbf{x}, t)$ that has value 1 if and only if s_{t-1} has the label SPEAKERNAME, s_t has the label OTHER, and the word x_t begins with a capital letter. The chief practical advantage of conditional models, in fact, is that we can include arbitrary highly-dependent features without needing to estimate their distribution, as would be required to learn a generative model.

Given fully-labeled training instances $\{(\mathbf{s}_j, \mathbf{x}_j)\}_{j=1}^{M}$, CRF training is usually performed by maximizing the penalized log likelihood

$$\ell(\Lambda) = \sum_j \sum_t \sum_k \lambda_k f_k(s_{j,t-1}, s_{j,t}, \mathbf{x}, t) - \sum_j \log Z(\mathbf{x}_j) - \sum_k \frac{\lambda_k^2}{2\sigma^2} \tag{4}$$

where the final term is a zero-mean Gaussian prior placed on parameters to avoid overfitting. Although this maximization cannot be done in closed form, it can be optimized numerically. Particularly effective are gradient-based methods that use approximate second-order information, such as conjugate gradient and limited-memory BFGS (Byrd et al., 1994). For more information on current training methods for CRFs, see Sha and Pereira (2003).

3 Dynamic CRFs

Dynamic conditional random fields (Sutton et al., 2004) extend linear-chain CRFs in the same way that dynamic Bayes nets (Dean & Kanazawa, 1989) extend HMMs. Rather than having a single monolithic state variable, DCRFs factorize the state at each time step by an undirected model.

Formally, DCRFs are the class of conditionally-trained undirected models that repeat structure and parameters over a sequence. If we denote by $\Phi_c(\mathbf{y}_{c,t}, \mathbf{x}_t)$ the repetition of clique c at time step t, then a DCRF defines the probability of a label sequence $\mathbf{s}$ given the input $\mathbf{x}$ as:

$$p(\mathbf{s}|\mathbf{x}) = \frac{\prod_t \Phi_c(\mathbf{y}_{c,t}, \mathbf{x}_t)}{Z(\mathbf{x})}, \tag{5}$$

where as before, the clique templates are parameterized in terms of input features as

$$\Phi_c(\mathbf{y}_{c,t}, \mathbf{x}_t) = \exp\left\{\sum_k \lambda_k f_k(\mathbf{y}_{c,t}, \mathbf{x}_t)\right\}. \tag{6}$$

Exact inference in DCRFs can be performed by forward-backward in the cross product state space, if the cross-product space is not so large as to be infeasible. Otherwise, approximate methods must be used; in our experience, loopy belief propagation is often effective in grid-shaped DCRFs. Even if inference is performed monolithically, however, a factorized state representation is still useful because it requires much fewer parameters than a fully-parameterized linear chain in the cross-product state space.

Sutton et al. (2004) introduced the *factorial CRF* (FCRF), in which the factorized state structure is a grid (Figure 1). FCRFs were originally applied to jointly performing interdependent language processing tasks, in particular part-of-speech tagging and noun-phrase chunking. The previous work on FCRFs used joint training, which requires a single training set that is jointly labeled for all tasks in the cascade. For many tasks such data is not readily available, for example, labeling syntactic parse trees for every new Web extraction task would be prohibitively expensive. In this paper, we train the subtasks separately, which allows us the freedom to use large, standard data sets for well-studied subtasks such as named-entity recognition.

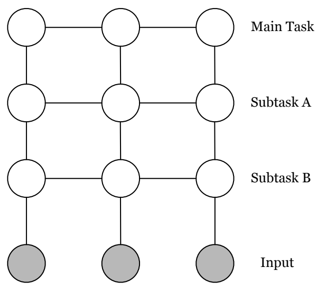


Figure 1: Graphical model for the jointly-decoded CRF. All of the pairwise cliques also have links to the observed input, although we omit these edges in the diagram for clarity.

4 Alternatives for Learning Transfer

In this section, we enumerate several classes of methods for learning transfer, based on the amount and type of interaction they allow between the tasks. The principal differences between methods are whether the individual tasks are performed separately in a cascade or jointly; whether a single prediction from the lower task is used, or several; and what kind of confidence information is shared between the subtasks.

The main types of transfer learning methods are:

1. *Cascaded training and testing.* This is the traditional approach in NLP, in which the single best prediction from the old task is used in the new task at training and test time. In this paper, we show that allowing richer interactions between the subtasks can benefit performance.

2. *Joint training and testing.* In this family of approaches, a single model is trained to perform all the subtasks at once. For example, in Caruana's work on multitask learning (Caruana, 1997), a neural network is trained to jointly perform multiple classification tasks, with hidden nodes that form a shared representation among the tasks. Jointly trained methods allow potentially the richest interaction between tasks, but can be expensive in both computation time required for training and in human effort required to label the joint training data.

   Exact inference in a jointly-trained model, such as forward-backward in an FCRF, implicitly considers all possible subtask predictions with confidence given by the model's probability of the prediction. However, for computational efficiency, we can use inference methods such as particle filtering and sparse message-passing (Pal et al., 2005), which communicate only a limited number of predictions between sections of the model.

3. *Joint testing with cascaded training.* Although a joint model over all the subtasks can have better performance, it is often much more expensive to train. One approach for reducing training time is cascaded training, which provides both computational efficiency and the ability to reuse large, standard training sets for the subtasks. At test time, though, the separately-trained models are combined into a single model, so that joint decoding can propagate information between the tasks.

   Even with cascaded training, it is possible to preserve some uncertainty in the subtask's predictions. Instead of using only a single subtask prediction for training the main task, the subtask can pass upwards a lattice of likely predictions, each of which is weighted by the model's confidence. This has the advantage of making the training procedure more similar to the joint testing procedure, in which all possible subtask predictions are considered.

In the next two sections, we describe and evaluate joint testing with cascaded training for transfer learning in linear-chain CRFs. At training time, only the best subtask prediction is used, without any confidence information. Even though this is perhaps the simplest joint-testing/cascaded-training method, we show that it still leads to a significant gain in accuracy.

5 Composition of CRFs

In this section we briefly describe how we combine individually-trained linear-chain CRFs using composition. For a series of N cascaded tasks, we train individual CRFs separately on each task, using the prediction of the previous CRF as a feature. We index the CRFs by i, so that the state of CRF i at time t is denoted s^i_t. Thus, the feature functions for CRF i are of the form $f^i_k(s^i_{t-1}, s^i_t, s^{i-1}_t, \mathbf{x}, t)$—that is, they depend not only on the observed input $\mathbf{x}$ and the transition $(s^i_{t-1} \to s^i_t)$ but

$w_t = w$
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t appears in list of first names, last names, honorifics, etc.
w_t appears to be part of a time followed by a dash
w_t appears to be part of a time preceded by a dash
w_t appears to be part of a date
$T_t = T$
$q_k(\mathbf{x}, t+\delta)$ for all k and $\delta \in [-4, 4]$

Table 1: Input features $q_k(\mathbf{x}, t)$ for the seminars data. In the above w_t is the word at position t, T_t is the POS tag at position t, w ranges over all words in the training data, and T ranges over all Penn Treebank part-of-speech tags. The "appears to be" features are based on hand-designed regular expressions that can span several tokens.

also on the state s_t^{i-1} of the previous transducer.

We also add all conjunctions of the input features and the previous transducer's state, for example, a feature that is 1 if the current state is SPEAKERNAME, the previous transducer predicted PERSONNAME, and the previous word is `Host:`.

To perform joint decoding at test time, we form the composition of the individual CRFs, viewed as finite-state transducers. That is, we define a new linear-chain CRF whose state space is the cross product of the states of the individual CRFs, and whose transition costs are the sum of the transition costs of the individual CRFs.

Formally, let $S^1, S^2, \ldots S^N$ be the state sets and $\Lambda^1, \Lambda^2, \ldots \Lambda^N$ the weights of the individual CRFs. Then the state set of the combined CRF is $\mathbf{S} = S^1 \times S^2 \times \ldots \times S^N$. We will denote weight k in an individual CRF i by λ_k^i and a single feature by $f_k^i(s_{t-1}^i, s_t^i, s_t^{i-1}, \mathbf{x}, t)$. Then for $\mathbf{s} \in \mathbf{S}$, the combined model is given by:

$$p(\mathbf{s}|\mathbf{x}) = \frac{\prod_t \exp\left\{\sum_{i=1}^N \sum_k \lambda_k^i f_k^i(s_{t-1}^i, s_t^i, s_t^{i-1}, \mathbf{x}, t)\right\}}{Z(\mathbf{x})}. \tag{7}$$

The graphical model for the combined model is the factorial CRF in Figure 1.

6 Experiments

6.1 Email Seminar Announcements

We evaluate joint decoding on a collection of 485 e-mail messages announcing seminars at Carnegie Mellon University, gathered by Freitag (1998). The messages are annotated with the seminar's starting time, ending time, location, and speaker. This data set has been the subject of much previous work using a wide variety of learning methods. Despite all this work, however, the best

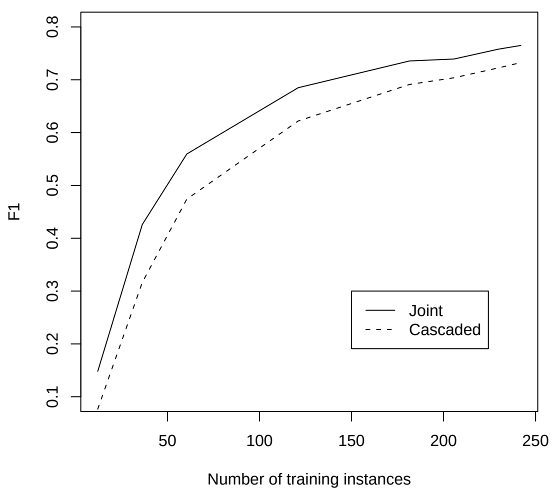


Figure 2: Learning curves for the seminars data set on the speaker field, averaged over 10-fold cross validation. Joint training performs equivalently to cascaded decoding with 25% more data.

reported systems have precision and recall on speaker names of only about 70%—too low to use in a practical system. This task is so challenging because the messages are written by many different people, who each have different ways of presenting the announcement information.

Because the task includes finding locations and person names, the output of a named-entity tagger is a useful feature. It is not a perfectly indicative feature, however, because many other kinds of person names appear in seminar announcements—for example, names of faculty hosts, departmental secretaries, and sponsors of lecture series. For example, the token *Host:* indicates strongly both that what follows is a person name, but that person is not the seminars' speaker.

Even so, named-entity predictions do improve performance on this task. We use the predictions from a CRF named-entity tagger that we trained on the standard CoNLL 2003 English data set. The CoNLL 2003 data set consists of newswire articles from Reuters labeled as either people, locations, organizations, or miscellaneous entities. It is much larger than the seminar announcements data set. While the named-entity data contains 203,621 tokens for training, the seminar announcements data set contains only slightly over 60,000 training tokens.

Previous work on the seminars data has used a one-field-per-document evaluation. That is, for each field, the CRF selects a single field value from its Viterbi path, and this extraction is counted as correct if it exactly matches any of the true field mentions in the document. We compute precision and recall following this convention, and report their harmonic mean F_1. As in the previous work,

System		stime	etime	location	speaker	overall
WHISK	(Soderland, 1999)	92.6	86.1	66.6	18.3	65.9
SRV	(Freitag, 1998)	98.5	77.9	72.7	56.3	76.4
HMM	(Frietag & McCallum, 1999)	98.5	62.1	78.6	76.6	78.9
RAPIER	(Califf & Mooney, 1999)	95.9	94.6	73.4	53.1	79.3
SNOW-IE	(Roth & Wen-tau Yih, 2001)	**99.6**	**96.3**	75.2	73.8	86.2
$(LP)^2$	(Ciravegna, 2001)	99.0	95.5	75.0	**77.6**	86.8
CRF (no transfer)	This paper	99.1	97.3	81.0	73.7	87.8
CRF (cascaded)	This paper	99.2	96.0	84.3	74.2	88.4
CRF (joint)	This paper	99.1	96.0	**85.3**	76.3	**89.2**

Table 2: Comparison of F_1 performance on the seminars data. Joint decoding performs significantly better than cascaded decoding. The overall column is the mean of the other four. (This table was adapted from Peshkin and Pfeffer (2003).)

we use 10-fold cross validation with a 50/50 training/test split. We use a spherical Gaussian prior on parameters with variance $\sigma^2 = 0.5$.

We evaluate whether joint decoding with cascaded training performs better than cascaded training and decoding. Table 2 compares cascaded and joint decoding for CRFs with other previous results from the literature.[1] The features we use are listed in Table 1. Although previous work has used very different feature sets, we include a no-transfer CRF baseline to assess the impact of transfer from the CoNLL data set. All the CRF runs used exactly the same features.

On the most challenging fields, location and speaker, cascaded transfer is more accurate than no transfer at all, and joint decoding is more accurate than cascaded decoding. In particular, for speaker, we see an error reduction of 8% by using joint decoding over cascaded. The difference in F1 between cascaded and joint decoding is statistically significant for speaker (paired t-test; $p = 0.017$) but only marginally significant for location ($p = 0.067$). Our results are competitive with previous work; for example, on location, the CRF is more accurate than any of the existing systems.

Examining the trained models, we can observe both errors made by the general-purpose named entity tagger, and how they can be corrected by considering the seminars labels. In newswire text, long runs of capitalized words are rare, often indicating the name of an entity. In email announcements, runs of capitalized words are common in formatted text blocks like:

```
Location:  Baker Hall
    Host:  Michael Erdmann
```

In this type of situation, the named entity tagger often mistakes *Host:* for the name of an entity, especially because the word preceding *Host* is also capitalized. On one of the cross-validated testing sets, of 80 occurrences of the word *Host:*, the named-entity tagger labels 52 as some kind of entity. When joint decoding is used, however, only 20 occurrences are labeled as entities. Recall that the joint model uses exactly the same weights as the cascaded model; the only difference is that the joint model takes into account information about the seminar labels when choosing named-entity labels. This is an example of how domain-specific information from the main task can improve performance on a more standard, general-purpose subtask.

[1]We omit one relevant paper (Peshkin & Pfeffer, 2003) because its evaluation method differs from all the other previous work.

$w_t = w$
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t is punctuation
w_t appears in list of first names, last names, honorifics, etc.
$q_k(\mathbf{x}, t + \delta)$ for all k and $\delta \in [-2, 2]$
Conjunction $q_k(\mathbf{x}, t)$ and $q_{k'}(\mathbf{x}, t)$ for all features k, k'
Conjunction $q_k(\mathbf{x}, t)$ and $q_{k'}(\mathbf{x}, t + 1)$ for all features k, k'

Table 3: Input features $q_k(\mathbf{x}, t)$ for the ACE named-entity data. In the above w_t is the word at position t, and w ranges over all words in the training data.

Figure 2 shows the difference in performance between joint and cascaded decoding as a function of training set size. Cascaded decoding with the full training set of 242 emails performs equivalently to joint decoding on only 181 training instances, a 25% reduction in the training set.

In summary, even with a simple cascaded training method on a well-studied data set, joint decoding performs better for transfer than cascaded decoding.

6.2 Entity Recognition

In this section we give results on joint decoding for transfer between two newswire data sets with similar but overlapping label sets. The Automatic Content Extraction (ACE) data set is another standard entity recognition data

	Transfer Type		
	none	cascaded	joint
Person name	81.0	86.9	87.3
Person nominal	34.9	36.1	42.4
Organization name	53.9	62.6	61.1
Organization nominal	33.7	35.3	40.8
GPE name	78.5	84.0	84.0
GPE nominal	51.2	54.1	59.2

Table 4: Comparison of F_1 performance between joint and cascaded training on the ACE entity recognition task. GPE means geopolitical entities, such as countries. Joint decoding helps most on the harder nominal (common noun) references. These results were obtained using a small subset of the training set.

set, containing 422 stories from newspaper, newswire, and broadcast news. Unlike the CoNLL entity recognition data set, in which only proper names of entities are annotated, the ACE data includes annotation both for named entities like *United States*, and also nominal mentions of entities like *the nation*. Thus, although the input text has similar distribution in the CoNLL NER and ACE data set, the label distributions are very different.

Current state-of-the-art systems for the ACE task (Florian et al., 2004) use the predictions of other named-entity recognizers as features, that is, they use cascaded transfer. In this experiment, we test whether the transfer between these datasets can be further improved using joint decoding. We train a CRF entity recognizer on the ACE dataset, with the output of a named-entity entity recognizer trained on the CoNLL 2003 English data set. The CoNLL recognizer is the same CRF as was used in the previous experiment. In these results, we use a subset of 10% of the ACE training data. Table 3 lists the features we use. Table 4 compares the results on some representative entity types. Again, cascaded decoding for transfer is better than no transfer at al, and joint decoding is better than cascaded decoding. Interestingly, joint decoding has most impact on the harder nominal references, showing marked improvement over the cascaded approach.

7 Related Work

Researchers have begun to accumulate experimental evidence that joint training and decoding yields better performance than the cascaded approach. As mentioned earlier, the original work on dynamic CRFs (Sutton et al., 2004) demonstrated improvement due to joint training in the domains of part-of-speech tagging and noun-phrase chunking. Also, Carreras and Marquez (Carreras & Màrquez, 2004) have obtained increased performance in clause finding by training a cascade of perceptrons to minimize a single global error function. Finally, Miller et al. (Miller et al., 2000) have combined entity recognition, parsing, and relation extraction into a jointly-trained single statistical parsing model that achieves improved performance on all the subtasks.

Part of the contribution of the current work is to suggest that joint decoding can be effective even when joint training is not possible because jointly-labeled data is unavailable. For example, Miller et al. report that they originally attempted to annotate newswire articles for all of parsing, relations, and named entities, but they stopped because the annotation was simply too expensive. Instead they hand-labeled relations only, assigning parse trees to the training set using a standard statistical parser, which is potentially less flexible than the cascaded training, because the model for main task is trained explicitly to match the noisy subtask predictions, rather than being free to correct them.

In the speech community, it is common to compose separately trained weighted finite-state transducers (Mohri et al., 2002) for joint decoding. Our method extends this work to conditional models. Ordinarily, higher-level transducers depend only on the output of the previous transducer: a transducer for the lexicon, for example, consumes only phonemes, not the original speech signal. In text, however, such an approach is not sensible, because there is simply not enough information in the named-entity labels, for example, to do extraction if the original words are discarded. In a conditional model, weights in higher-level transducers are free to depend on arbitrary features of the original input without any additional complexity in the finite-state structure.

Finally, stacked sequential learning (Cohen & Carvalho, 2005) is another potential method for combining the results of the subtask transducers. In this general meta-learning method for sequential classification, first a base classifier predicts the label at each time step, and then a higher-level classifier makes the final prediction, including as features a window of predictions from the base classifier. For transfer learning, this would correspond to having an independent base model for each subtask (e.g., independent CRFs for named-entity and seminars), and then having a higher-level CRF that includes as a feature the predictions from the base models.

8 Conclusion

In this paper we have shown that joint decoding improves transfer between interdependent NLP tasks, even when the old task is named-entity recognition, for which highly accurate systems exist. The rich features afforded by a conditional model allow the new task to influence the predictions of the old task, an effect that is only possible with joint decoding.

It is now common for researchers to publicly release trained models for standard tasks such as part-of-speech tagging, named-entity recognition, and parsing. This paper has implications for how such standard tools are packaged. Our results suggest that off-the-shelf NLP tools will need not only to provide a single-best prediction, but also to be engineered so that they can easily communicate distributions over predictions to models for higher-level tasks.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grants #IIS-0326249 and #IIS-0427594, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Byrd, R. H., Nocedal, J., & Schnabel, R. B. (1994). Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Program.*, *63*, 129–156.

Califf, M. E., & Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)* (pp. 328–334).

Carreras, X., & Marquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. *Proceedings of CoNLL-2004*.

Carreras, X., & Màrquez, L. (2004). Online learning via global feedback for phrase recognition. In S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Ciravegna, F. (2001). Adaptive information extraction from text by rule induction and generalisation. *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*.

Cohen, W. W., & Carvalho, V. R. (2005). Stacked sequential learning. *International Joint Conference on Artificial Intelligence* (pp. 671–676).

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, *5(3)*, 142–150.

Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, N., Roukos, S., & Zhang, T. (2004). A statistical model for multilingual entity detection and tracking. *In* HLT/NAACL 2004.

Freitag, D. (1998). *Machine learning for information extraction in informal domains*. Doctoral dissertation, Carnegie Mellon University.

Frietag, D., & McCallum, A. (1999). Information extraction with HMMs and shrinkage. *AAAI Workshop on Machine Learning for Information Extraction*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*.

Miller, S., Fox, H., Ramshaw, L. A., & Weischedel, R. M. (2000). A novel use of statistical parsing to extract information from text. *ANLP 2000* (pp. 226–233).

Mohri, M., Pereira, F., & Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, *16*, 69–88.

Pal, C., Sutton, C., & McCallum, A. (2005). *Fast inference and learning with sparse belief propagation* (Technical Report IR-433). Center for Intelligent Information Retrieval, University of Massachusetts.

Peshkin, L., & Pfeffer, A. (2003). Bayesian information extraction network. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Roth, D., & Wen-tau Yih (2001). Relational learning via propositional algorithms: An information extraction case study. *International Joint Conference on Artificial Intelligence* (pp. 1257–1263).

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *Proceedings of HLT-NAACL 2003*.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. *Machine Learning*, 233–272.

Sutton, C., Rohanimanesh, K., & McCallum, A. (2004). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*.

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *Proceedings of CoNLL-2003* (pp. 142–147). Edmonton, Canada.

Translating with non-contiguous phrases

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada
Xerox Research Centre Europe
FirstName.FamilyName@xrce.xerox.com

Philippe Langlais
RALI/DIRO Université de Montréal
felipe@iro.umontreal.ca

Arne Mauser
RWTH Aachen University
arne.mauser@rwth-aachen.de

Abstract

This paper presents a phrase-based statistical machine translation method, based on non-contiguous phrases, i.e. phrases with *gaps*. A method for producing such phrases from a word-aligned corpora is proposed. A statistical translation model is also presented that deals such phrases, as well as a training method based on the maximization of translation accuracy, as measured with the NIST evaluation metric. Translations are produced by means of a beam-search decoder. Experimental results are presented, that demonstrate how the proposed method allows to better generalize from the training data.

1 Introduction

Possibly the most remarkable evolution of recent years in statistical machine translation is the step from word-based models to phrase-based models (Och et al., 1999; Marcu and Wong, 2002; Yamada and Knight, 2002; Tillmann and Xia, 2003). While in traditional word-based statistical models (Brown et al., 1993) the atomic unit that translation operates on is the word, phrase-based methods acknowledge the significant role played in language by multi-word expressions, thus incorporating in a statistical framework the insight behind Example-Based Machine Translation (Somers, 1999).

However, Phrase-based models proposed so far only deal with multi-word units that are sequences of contiguous words on both the source and the target side. We propose here a model designed to deal with multi-word expressions that need not be contiguous in either or both the source and the target side.

The rest of this paper is organised as follows. Section 2 provides motivations, definition and extraction procedure for non-contiguous phrases. The log-linear conditional translation model we adopted is the object of Section 3; the method used to train its parameters is described in Section 4. Section 5 briefly describes the decoder. The experiments we conducted to asses the effectiveness of using non-contiguous phrases are presented in Section 6.

2 Non-contiguous phrases

Why should it be a good thing to use phrases composed of possibly non-contiguous sequences of words? In doing so we expect to improve translation quality by better accounting for additional linguistic phenomena as well as by extending the effect of contextual semantic disambiguation and example-based translation inherent in phrase-based MT. An example of a phenomenon best described using non-contiguous units is provided by English phrasal verbs. Consider the sentence “Mary *switches* her table lamp *off*”. Word-based statistical models would be at odds when selecting the appropriate translation of the verb. If French were the target language, for instance, corpus evidence would come from both examples in which “switch” is translated as “allumer” (to switch on) and as “éteindre” (to switch off). If many-to-one word alignments are not allowed from English to French, as it is usually the

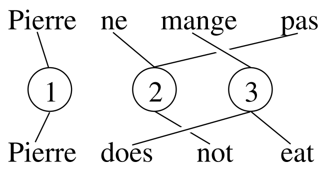


Figure 1: An example of a complex alignment associated with different syntax for negation in English and French.

case, then the best thing a word-based model could do in this case would be to align "off" to the empty word and hope to select the correct translation from "switch" only, basically a 50-50 bet. While handling inseparable phrasal verbs such as "to run out" correctly, previously proposed phrase-based models would be helpless in this case. A comparable behavior is displayed by German separable verbs. Moreover, non-contiguous linguistic units are not limited to verbs. Negation is formed, in French, by inserting the words "ne" and "pas" before and after a verb respectively. So, the sentence "Pierre ne mange pas" and its English translation display a complex word-level alignment (Figure 1) current models cannot account for.

Flexible idioms, allowing for the insertion of linguistic material, are other phenomena best modeled with non-contiguous units.

2.1 Definition and library construction

We define a *bi-phrase* as a pair comprising a *source phrase* and a *target phrase*: $b = \langle \tilde{s}, \tilde{t} \rangle$. Each of the source and target phrases is a sequence of words and gaps (indicated by the symbol $\diamond$); each gap acts as a placeholder for exactly one unspecified word. For example, $\tilde{w} = w_1 w_2 \diamond w_3 \diamond\diamond\, w_4$ is a phrase of length 7, made up of two contiguous words w_1 and w_2, a first gap, a third word w_3, two consecutive gaps and a final word w_4. To avoid redundancy, phrases may not begin or end with a gap. If a phrase does not contain any gaps, we say it is *contiguous*; otherwise it is *non-contiguous*. Likewise, a bi-phrase is said to be *contiguous* if both its phrases are contiguous.

The translation of a source sentence s is produced by combining together bi-phrases so as to cover the source sentence, and produce a well-formed target-language sentence (i.e. without gaps). A complete translation for s can be described as an ordered sequence of bi-phrases $b_1 ... b_K$. When piecing together the final translation, the target-language portion $\tilde{t}_1$ of the first bi-phrase b_1 is first layed down, then each subsequent $\tilde{t}_k$ is positioned on the first "free" position in the target language sentence, i.e. either the leftmost gap, or the right end of the sequence. Figure 2 illustrates this process with an example.

To produce translations, our approach therefore relies on a collection of bi-phrases, what we call a *bi-phrase library*. Such a library is constructed from a corpus of existing translations, aligned at the word level.

Two strategies come to mind to produce non-contiguous bi-phrases for these libraries. The first is to align the words using a "standard" word alignement technique, such as the *Refined Method* described in (Och and Ney, 2003) (the intersection of two IBM Viterbi alignments, forward and reverse, enriched with alignments from the union) and then generate bi-phrases by combining together individual alignments that co-occur in the same pair of sentences. This is the strategy that is usually adopted in other phrase-based MT approaches (Zens and Ney, 2003; Och and Ney, 2004). Here, the difference is that we are not restricted to combinations that produce strictly contiguous bi-phrases.

The second strategy is to rely on a word-alignment method that naturally produces many-to-many alignments between non-contiguous words, such as the method described in (Goutte et al., 2004). By means of a matrix factorization, this method produces a parallel partition of the two texts, seen as sets of word tokens. Each token therefore belongs to one, and only one, subset within this partition, and corresponding subsets in the source and target make up what are called *cepts*. For example, in Figure 1, these cepts are represented by the circles numbered 1, 2 and 3; each cept thus connects word tokens in the source and the target, regardless of position or contiguity. These cepts naturally constitute bi-phrases, and can be used directly to produce a bi-phrase library.

Obviously, the two strategies can be combined, and it is always possible to produce increasingly large and complex bi-phrases by combining together co-occurring bi-phrases, contiguous or not. One problem with this approach, however, is that the resulting libraries can become very large. With con-

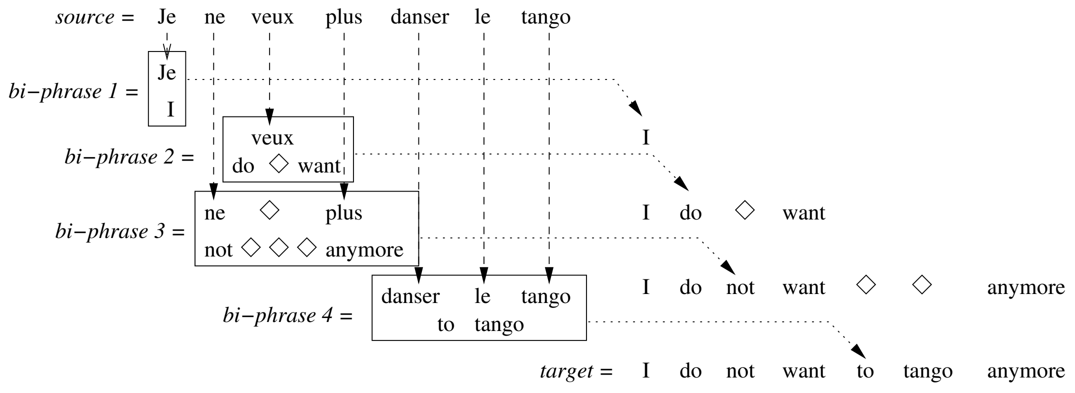

source = Je ne veux plus danser le tango
bi−phrase 1 = Je / I
bi−phrase 2 = veux / do ◇ want
bi−phrase 3 = ne ◇ plus / not ◇ ◇ ◇ anymore
bi−phrase 4 = danser le tango / to tango
I
I do ◇ want
I do not want ◇ ◇ anymore
target = I do not want to tango anymore

Figure 2: Combining bi-phrases to produce a translation.

tiguous phrases, the number of bi-phrases that can be extracted from a single pair of sentences typically grows quadratically with the size of the sentences; with non-contiguous phrases, however, this growth is exponential. As it turns out, the number of available bi-phrases for the translation of a sentence has a direct impact on the time required to compute the translation; we will therefore typically rely on various filtering techniques, aimed at keeping only those bi-phrases that are more likely to be useful. For example, we may retain only the most frequently observed bi-phrases, or impose limits on the number of cepts, the size of gaps, etc.

3 The Model

In statistical machine translation, we are given a source language input $s_1^J = s_1 ... s_J$, and seek the target-language sentence $t_1^I = t_1 ... t_I$ that is its most likely translation:

$$\hat{t}_1^I = \text{argmax}_{t_1^I} Pr(t_1^I | s_1^J) \tag{1}$$

Our approach is based on a direct approximation of the posterior probability $Pr(t_1^I | s_1^J)$, using a log-linear model:

$$Pr(t_1^I | s_1^J) = \frac{1}{Z_{s_1^J}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(t_1^I, s_1^J)\right)$$

In such a model, the contribution of each *feature function* h_m is determined by the corresponding model parameter λ_m; $Z_{s_1^J}$ denotes a normalization constant. This type of model is now quite widely used for machine translation (Tillmann and Xia, 2003; Zens and Ney, 2003)[1].

Additional variables can be introduced in such a model, so as to account for hidden characteristics, and the feature functions can be extended accordingly. For example, our model must take into account the actual set of bi-phrases that was used to produce this translation:

$$Pr(t_1^I, b_1^K | s_1^J) = \frac{1}{Z_{s_1^J}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(t_1^I, s_1^J, b_1^K)\right)$$

Our model currently relies on seven feature functions, which we describe here.

- The *bi-phrase* feature function h_{bp}: it represents the probability of producing t_1^I using some set of bi-phrases, under the assumption that each source phrase produces a target phrase independently of the others:

$$h_{bp}(t_1^I, s_1^J, b_1^K) = \sum_{k=1}^{K} \log Pr(\tilde{t}_k | \tilde{s}_k) \tag{2}$$

  Individual bi-phrase probabilities $Pr(\tilde{t}_k | \tilde{s}_k)$ are estimated based on occurrence counts in the word-aligned training corpus.

- The *compositional bi-phrase* feature function h_{comp}: this is introduced to compensate for

[1]Recent work from Chiang (Chiang, 2005) addresses similar concerns to those motivating our work by introducing a Synchronous CFG for bi-phrases. If on one hand SCFGs allow to better control the order of the material inserted in the gaps, on the other gap size does not seem to be taken into account, and phrase dovetailing such as the one involving "do ◇want" and "not ◇◇◇anymore" in Fig. 2 is disallowed.

h_{bp}'s strong tendency to overestimate the probability of rare bi-phrases; it is computed as in equation (2), except that bi-phrase probabilities are computed based on individual word translation probabilities, somewhat as in IBM model 1 (Brown et al., 1993):

$$Pr(\tilde{t}|\tilde{s}) = \frac{1}{|\tilde{s}|^{|\tilde{t}|}} \prod_{t \in \tilde{t}} \sum_{s \in \tilde{s}} Pr(t|s)$$

- The *target language* feature function h_{tl}: this is based on a N-gram language model of the target language. As such, it ignores the source language sentence and the decomposition of the target into bi-phrases, to focus on the actual sequence of target-language words produced by the combination of bi-phrases:

$$h_{tl}(t_1^I, s_1^J, b_1^K) = \sum_{i=1}^{I} \log Pr(t_i | t_{i-N+1}^{i-1})$$

- The *word-count* and *bi-phrase count* feature functions h_{wc} and h_{bc}: these control the length of the translation and the number of bi-phrases used to produce it:

$$h_{wc}(t_1^I, s_1^J, b_1^K) = I \quad h_{bc}(t_1^I, s_1^J, b_1^K) = K$$

- The *reordering* feature function $h_{reord}(t_1^I, s_1^J, b_1^K)$: it measures the amount of reordering between bi-phrases of the source and target sentences.

- the *gap count* feature function h_{gc}: It takes as value the total number of gaps (source and target) within the bi-phrases of b_1^K, thus allowing the model some control over the nature of the bi-phrases it uses, in terms of the discontiguities they contain.

4 Parameter Estimation

The values of the λ parameters of the log-linear model can be set so as to optimize a given criterion. For instance, one can maximize the likelyhood of some set of training sentences. Instead, and as suggested by Och (2003), we chose to maximize directly the quality of the translations produced by the system, as measured with a machine translation evaluation metric.

Say we have a set of source-language sentences S. For a given value of λ, we can compute the set of corresponding target-language translations T. Given a set of *reference* ("gold-standard") translations R for S and a function $E(T, R)$ which measures the "error" in T relative to R, then we can formulate the parameter estimation problem as[2]:

$$\hat{\lambda} = \text{argmin}_{\lambda} E(T, R)$$

As pointed out by Och, one notable difficulty with this approach is that, because the computation of T is based on an argmax operation (see eq. 1), it is not continuous with regard to λ, and standard gradient-descent methods cannot be used to solve the optimization. Och proposes two workarounds to this problem: the first one relies on a direct optimization method derived from Powell's algorithm; the second introduces a smoothed (continuous) version of the error function $E(T, R)$ and then relies on a gradient-based optimization method.

We have opted for this last approach. Och shows how to implement it when the error function can be computed as the sum of errors on individual sentences. Unfortunately, this is not the case for such widely used MT evaluation metrics as BLEU (Papineni et al., 2002) and NIST (Doddington, 2002). We show here how it can be done for NIST; a similar derivation is possible for BLEU.

The NIST evaluation metric computes a weighted n-gram precision between T and R, multiplied by a factor $B(S, T, R)$ that penalizes short translations. It can be formulated as:

$$B(S, T, R) \times \sum_{n=1}^{N} \frac{\sum_{s \in S} I_n(t_s, r_s)}{\sum_{s \in S} C_n(t_s)} \tag{3}$$

where N is the largest n-gram considered (usually $N = 4$), $I_n(t_s, r_s)$ is a weighted count of common n-grams between the target (t_s) and reference (r_s) translations of sentence s, and $C_n(t_s)$ is the total number of n-grams in t_s.

To derive a version of this formula that is a continuous function of λ, we will need multiple translations $t_{s,1}, ..., t_{s,K}$ for each source sentence s. The general idea is to weight each of these translations

[2]For the sake of simplicity, we consider a single reference translation per source sentence, but the argument can easily be extended to multiple references.

by a factor $w(\lambda, s, k)$, proportional to the score $m_\lambda(t_{s,k}|s)$ that $t_{s,k}$ is assigned by the log-linear model for a given λ:

$$w(\lambda, s, k) = \left[\frac{m_\lambda(t_{s,k}|s)}{\sum_{k'} m_\lambda(t_{s,k'}|s)} \right]^\alpha$$

where α is the *smoothing factor*. Thus, in the smoothed version of the NIST function, the term $I_n(t_s, r_s)$ in equation (3) is replaced by $\sum_k w(\lambda, s, k) I_n(t_{s,k}, r_s)$, and the term $C_n(t_s)$ is replaced by $\sum_k w(\lambda, s, k) C_n(t_{s,k})$. As for the brevity penalty factor $B(S, T, R)$, it depends on the total length of translation T, i.e. $\sum_s |t_s|$. In the smoothed version, this term is replaced by $\sum_s \sum_k w(\lambda, s, k)|t_{s,k}|$. Note that, when $\alpha \to \infty$, then $w(\lambda, s, k) \to 0$ for all translations of s, except the one for which the model gives the highest score, and so the smooth and normal NIST functions produce the same value. In practice, we determine some "good" value for α by trial and error (5 works fine).

We thus obtain a scoring function for which we can compute a derivative relative to λ, and which can be optimized using gradient-based methods. In practice, we use the *OPT++* implementation of a quasi-Newton optimization (Meza, 1994). As observed by Och, the smoothed error function is not convex, and therefore this sort of minimum-error rate training is quite sensitive to the initialization values for the λ parameters. Our approach is to use a random set of initializations for the parameters, perform the optimization for each initialization, and select the model which gives the overall best performance.

Globally, parameter estimation proceeds along these steps:

1. Initialize the training set: using random parameter values λ_0, for each source sentence of some given set of sentences S, we compute multiple translations. (In practice, we use the M-best translations produced by our decoder; see Section 5).

2. Optimize the parameters: using the method described above, we find λ that produces the best smoothed NIST score on the training set.

3. Iterate: we then re-translate the sentences of S with this new λ, combine the resulting multiple translations with those already in the training set, and go back to step 2.

Steps 2 and 3 can be repeated until the smooothed NIST score does not increase anymore[3].

5 Decoder

We implemented a version of the beam-search stack decoder described in (Koehn, 2003), extended to cope with non-contiguous phrases. Each translation is the result of a sequence of *decisions*, each of which involves the selection of a bi-phrase and of a target position. The final result is obtained by combining decisions, as in Figure 2. *Hypotheses*, corresponding to partial translations, are organised in a sequence of priority stacks, one for each number of source words covered. Hypotheses are extended by filling the first available uncovered position in the target sentence; each extended hypotheses is then inserted in the stack corresponding to the updated number of covered source words. Each hypothesis is assigned a score which is obtained as a combination of the actual feature function values and of admissible heuristics, adapted to deal with gaps in phrases, estimating the future cost for completing a translation. Each stack undergoes both threshold and histogram pruning. Whenever two hypotheses are indistinguishable as far as the potential for further extension is concerned, they are merged and only the highest-scoring is further extended. Complete translations are eventually recovered in the "last" priority stack, i.e. the one corresponding to the total number of source words: the best translation is the one with the highest score, and that does not have any remaining gaps in the target.

6 Evaluation

We have conducted a number of experiments to evaluate the potential of our approach. We were particularly interested in assessing the impact of non-contiguous bi-phrases on translation quality, as well as comparing the different bi-phrase library contruction strategies evoked in Section 2.1.

[3]It can be seen that, as the set of possible translations for S stabilizes, we eventually reach a point where the procedure converges to a maximum. In practice, however, we can usually stop much earlier.

6.1 Experimental Setting

All our experiments focused exclusively on French to English translation, and were conducted using the Aligned Hansards of the 36th Parliament of Canada, provided by the Natural Language Group of the USC Information Sciences Institute, and edited by Ulrich Germann. From this data, we extracted three distinct subcorpora, which we refer to as the *bi-phrase-building set*, the *training set* and the *test set*. These were extracted from the so-called *training*, *test-1* and *test-2* portions of the Aligned Hansard, respectively. Because of efficiency issues, we limited ourselves to source-language sentences of 30 words or less. More details on the evaluation data is presented in Table 1[4].

6.2 Bi-phrase Libraries

From the bi-phrase-building set, we built a number of libraries. A first family of libraries was based on a word alignment "A", produced using the *Refined method* described in (Och and Ney, 2003) (combination of two IBM-Viterbi alignments): we call these the A libraries. A second family of libraries was built using alignments "B" produced with the method in (Goutte et al., 2004): these are the B libraries. The most notable difference between these two alignments is that B contains "native" non-contiguous bi-phrases, while A doesn't.

Some libraries were built by simply extracting the cepts from the alignments of the bi-phrase-building corpus: these are the A^1 and B^1 libraries, and variants. Other libraries were obtained by combining cepts that co-occur within the same pair of sentences, to produce "composite" bi-phrases. For instance, the A^2 libraries contain combinations of 1 or 2 cepts from alignment A; B^3 contains combinations of 1, 2 or 3 cepts, etc.

Some libraries were built using a "gap-size" filter. For instance library A^2-g3 contains those bi-phrases obtained by combining 1 or 2 cepts from alignment A, and in which neither the source nor the target phrase contains more than 3 gaps. In particular, library B^1-g0 does not contain any non-contiguous bi-phrases.

[4]Preliminary experiments on different data sets allowed us to establish that 800 sentences constituted an acceptable size for estimating model parameters. With such a corpus, the estimation procedure converges after just 2 or 3 iterations.

Finally, all libraries were subjected to the same two filtering procedures: the first excludes all bi-phrases that occur only once in the training corpus; the second, for any given source-language phrase, retains only the 20 most frequent target-language equivalents. While the first of these filters typically eliminates a large number of entries, the second only affects the most frequent source phrases, as most phrases have less than 20 translations.

6.3 Experiments

The parameters of the model were optimized independantly for each bi-phrase library. In all cases, we performed only 2 iterations of the training procedure, then measured the performance of the system on the test set in terms of the NIST and BLEU scores against one reference translation. As a point of comparison, we also trained an IBM-4 translation model with the *GIZA++* toolkit (Och and Ney, 2000), using the combined *bi-phrase building* and *training* sets, and translated the test set using the *ReWrite* decoder (Germann et al., 2001)[5].

Table 2 describes the various libraries that were used for our experiments, and the results obtained for each.

System/library	bi-phrases	NIST	BLEU
ReWrite		6.6838	0.3324
A^1	238 K	6.6695	0.3310
A^2-g0	642 K	6.7675	0.3363
A^2-g3	4.1 M	6.7068	0.3283
B^1-g0	193 K	6.7898	0.3369
B^1	267 K	6.9172	0.3407
B^2-g0	499 K	6.7290	0.3391
B^2-g3	3.3 M	6.9707	0.3552
B^1-g1	206 K	6.8979	0.3441
B^1-g2	213 K	6.9406	0.3454
B^1-g3	218 K	6.9546	0.3518
B^1-g4	222 K	6.9527	0.3423

Table 2: Bi-phrase libraries and results

The top part of the table presents the results for the A libraries. As can be seen, library A^1 achieves approximately the same score as the baseline system; this is expected, since this library is essentially

[5]Both the *ReWrite* and our own system relied on a trigram language model trained on the English half of the bi-phrase building set.

Subset	sentences	source words	target words
bi-phrase-building set	931,000	17.2M	15.2M
training set	800	11,667	10,601
test set	500	6726	6041

Table 1: Data sets.

made up of one-to-one alignments computed using IBM-4 translation models. Adding contiguous bi-phrases obtained by combining pairs of alignments does gain us some mileage (+0.1 NIST)[6]. Again, this is consistent with results observed with other systems (Tillmann and Xia, 2003). However, the addition of non-contiguous bi-phrases (A^2-g3) does not seem to help.

The middle part of Table 2 presents analogous results for the corresponding B libraries, plus the B^1-g0 library, which contains only those cepts from the B alignment that are contiguous. Interestingly, in the experiments reported in (Goutte et al., 2004), alignment method B did not compare favorably to A under the widely used *Alignment Error Rate* (AER) metric. Yet, the B^1-g0 library performs better than the analogous A^1 library on the translation task. This suggests that AER may not be an appropriate metric to measure the potential of an alignment for phrase-based translation.

Adding non-contiguous bi-phrases allows another small gain. Again, this is interesting, as it suggests that "native" non-contiguous bi-phrases are indeed useful for the translation task, i.e. those non-contiguous bi-phrases obtained directly as cepts in the B alignment.

Surprisingly, however, combining cepts from the B alignment to produce contiguous bi-phrases (B^2-G0) does not turn out to be fruitful. Why this is so is not obvious and, certainly, more experiments would be required to establish whether this tendency continues with larger combinations (B^3-g0, B^4-g0...). Composite non-contiguous bi-phrases produced with the B alignments (B^2-g3) seem to bring improvements with regard to "basic" bi-phrases (B_1), but it is not clear whether these are significant.

[6]While the differences in scores in these and other experiments are relatively small, we believe them to be significant, as they have been confirmed systematically in other experiments and, in our experience, by visual inspection of the translations.

Visual examination of the B^1 library reveals that many non-contiguous bi-phrases contain long-spanning phrases (i.e. phrases containing long sequences of gaps). To verify whether or not these were really useful, we tested a series of B^1 libraries with different gap-size filters. It must be noted that, because of the final histogram filtering we apply on libraries (retain only the 20 most frequent translations of any source phrase), library B^1-g1 is not a strict subset of B^1-g2. Therefore, filtering on gap-size usually represents a tradeoff between more frequent long-spanning bi-phrases and less frequent short-spanning ones.

The results of these experiments appear in the lower part of Table 2. While the differences in score are small, it seems that concentrating on bi-phrases with 3 gaps or less affords the best compromise. For small libraries such as those under consideration here, this sort of filtering may not be very important. However, for higher-order libraries (B^2, B^3, etc.) it becomes crucial, because it allows to control the exponential growth of the libraries.

7 Conclusions

In this paper, we have proposed a phrase-based statistical machine translation method based on non-contiguous phrases. We have also presented a estimation procedure for the parameters of a log-linear translation model, that maximizes a smooth version of the NIST scoring function, and therefore lends itself to standard gradient-based optimization techniques.

From our experiments with these new methods, we essentially draw two conclusions. The first and most obvious is that non-contiguous bi-phrases can indeed be fruitful in phrase-based statistical machine translation. While we are not yet able to characterize which bi-phrases are most helpful, some of those that we are currently capable of extracting are well suited to cover some short-distance phenomena.

The second conclusion is that alignment quality is crucial in producing good translations with phrase-based methods. While this may sound obvious, our experiments shed some light on two specific aspects of this question. The first is that the alignment method that produces the most useful bi-phrases need not be the one with the best *alignment error rate* (AER). The second is that, depending on the alignments one starts with, constructing increasingly large bi-phrases does not necessarily lead to better translations. Some of our best results were obtained with relatively small libraries (just over 200,000 entries) of short bi-phrases. In other words, it's not how many bi-phrases you have, it's how good they are. This is the line of research that we intend to pursue in the near future.

Acknowledgments

The authors are grateful to the anonymous reviewers for their useful suggestions. [7]

References

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.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270, Ann Arbor, Michigan.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. ARPA Workshop on Human Language Technology*.

U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada. 2001. Fast Decoding and Optimal Decoding for Machine Translation. In *Proceedings of ACL 2001*, Toulouse, France.

Cyril Goutte, Kenji Yamada, and Eric Gaussier. 2004. Aligning words using matrix factorisation. In *Proc. ACL'04*, pages 503–510.

Philipp Koehn. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California.

[7]This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP 02)*, Philadelphia, PA.

J. C. Meza. 1994. OPT++: An Object-Oriented Class Library for Nonlinear Optimization. Technical Report SAND94-8225, Sandia National Laboratories, Albuquerque, USA, March.

F. J. Och and H. Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL 2000*, pages 440–447, Hongkong, China, October.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VCL 99)*, College Park, MD.

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *ACL'03: 41st Ann. Meet. of the Assoc. for Computational Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, USA.

Harold Somers. 1999. Review Article: Example-based Machine Translation. *Machine Translation*, 14:113–157.

Christoph Tillmann and Fei Xia. 2003. A phrase-based unigram model for statistical machine translation. In *Proc. of the HLT-NAACL 2003 Conference*, Edmonton, Canada.

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)*, Philadelphia, PA.

Richard Zens and Hermann Ney. 2003. Improvements in Phrase-Based Statistical Machine Translation. In *Proc. of the HLT-NAACL 2003 Conference*, Edmonton, Canada.

Word-Level Confidence Estimation for Machine Translation using Phrase-Based Translation Models

Nicola Ueffing and Hermann Ney
Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
{ueffing,ney}@informatik.rwth-aachen.de

Abstract

Confidence measures for machine translation is a method for labeling each word in an automatically generated translation as correct or incorrect. In this paper, we will present a new approach to confidence estimation which has the advantage that it does not rely on system output such as N-best lists or word graphs as many other confidence measures do. It is, thus, applicable to any kind of machine translation system.

Experimental evaluation has been performed on translation of technical manuals in three different language pairs. Results will be presented for different machine translation systems to show that the new approach is independent of the underlying machine translation system which generated the translations. To the best of our knowledge, the performance of the new confidence measure is better than that of any existing confidence measure.

1 Introduction

The work presented in this paper deals with confidence estimation for machine translation (MT). Since sentences produced by a machine translation system are often incorrect but may contain correct parts, a method for identifying those correct parts and finding possible errors is desirable. For this purpose, each word in the generated target sentence is assigned a value expressing the confidence that it is correct.

Confidence measures have been extensively studied for speech recognition, but are not well known in other areas. Only recently have researchers started to investigate confidence measures for machine translation (Blatz et al., 2004; Gandrabur and Foster, 2003; Quirk, 2004; Ueffing et al., 2003).

We apply word confidence measures in MT as follows: For a given translation generated by a machine translation system, we determine a confidence value for each word and compare it to a threshold. All words whose confidence is above this threshold are tagged as correct and all others are tagged as incorrect translations. The threshold is optimized on a distinct development set beforehand.

Possible applications for confidence measures include

- post-editing, where words with low confidence could be marked as potential errors,
- improving translation prediction accuracy in trans-type-style interactive machine translation (Gandrabur and Foster, 2003; Ueffing and Ney, 2005),
- combining output from different machine translation systems: hypotheses with low confidence can be discarded before selecting one of the system translations (Akiba et al., 2004), or the word confidence scores can be used for generating new hypotheses from the output of different systems (Jayaraman and Lavie, 2005), or the sentence confidence value can be employed for re-ranking (Blatz et al., 2003).

In this paper, we will present several approaches to word-level confidence estimation and develop a new phrase-based confidence measure which is independent of the machine translation system which generated the translation. The paper is organized as follows: In section 2, we will briefly review the statistical approach to machine translation. The phrase-based translation system, which serves as basis for the new confidence measure, will be presented in section 2.2. Section 3 will give an overview of related work on confidence estimation for statistical machine translation (SMT). In section 4, we will describe methods for confidence estimation which make use of SMT system output such as word graphs and N-best lists. In section 5, we will present the new phrase-based confidence measure. Section 6 contains a short description of an IBM-1 based confidence measure to which we will compare the other measures. Experimental evaluation and comparison of the different confidence measures will be shown in section 7, and section 8 will conclude the paper.

2 Statistical machine translation

2.1 General

In statistical machine translation, the translation is modeled as a decision process: Given a source string $f_1^J = f_1 \ldots f_j \ldots f_J$, we seek the target string $e_1^I = e_1 \ldots e_i \ldots e_I$ with maximal posterior probability:

$$\begin{aligned} \hat{e}_1^{\hat{I}} &= \operatorname*{argmax}_{I,e_1^I} \left\{ Pr(e_1^I \mid f_1^J) \right\} \qquad (1) \\ &= \operatorname*{argmax}_{I,e_1^I} \left\{ Pr(f_1^J \mid e_1^I) \cdot Pr(e_1^I) \right\} \end{aligned}$$

Through this decomposition of the probability, we obtain two knowledge sources: the translation model $Pr(f_1^J \mid e_1^I)$ and the language model $Pr(e_1^I)$. Both of them can be modeled independently of each other. The translation model is responsible for linking the source string f_1^J and the target string e_1^I, i.e. it captures the semantics of the sentence. The target language model captures the well-formedness or the syntax in the target language. Nowadays, most of the state-of-the-art SMT systems are based on bilingual phrases (Bertoldi et al., 2004; Koehn et al., 2003; Och and Ney, 2004; Tillmann, 2003; Vogel et al., 2004; Zens and Ney, 2004). Note that those phrases are sequences of words in the two languages and not necessarily phrases in the linguistic sense. A more detailed description of a phrase-based approach to statistical machine translation will be given in section 2.2.

2.2 Review of phrase-based translation system

For the confidence measures which will be introduced in section 5, we use a state-of-the-art phrase-based approach as described in (Zens and Ney, 2004). The key elements of this translation approach are bilingual phrases, i.e. pairs of source and target language phrases where a phrase is simply a contiguous sequence of words. These bilingual phrases are extracted from a word-aligned bilingual training corpus.

We will present the equations for a monotone search here in order to keep the equations simple. Let (j_0^K, i_0^K) be a segmentation of the source sentence into phrases, with the corresponding (bilingual) phrase pairs $(\tilde{f}_k, \tilde{e}_k) = (f_{j_{k-1}+1}^{j_k}, e_{i_{k-1}+1}^{i_k}), k = 1, \ldots, K$. The phrase-based approach to SMT is then expressed by the following equation:

$$\begin{aligned} \hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{j_0^K, i_0^K, I, e_1^I} & \left\{ \prod_{i=1}^{I} \left[c_1 \cdot p(e_i \mid e_{i-2}^{i-1})^{\lambda_1} \right] \right. \qquad (2) \\ & \cdot \prod_{k=1}^{K} \left[c_2 \cdot \; p(\tilde{f}_k \mid \tilde{e}_k)^{\lambda_2} \cdot p(\tilde{e}_k \mid \tilde{f}_k)^{\lambda_3} \right. \\ & \left. \left. \cdot \prod_{j=j_{k-1}+1}^{j_k} p(f_j \mid \tilde{e}_k)^{\lambda_4} \cdot \prod_{i=i_{k-1}+1}^{i_k} p(e_i \mid \tilde{f}_k)^{\lambda_5} \right] \right\}, \end{aligned}$$

where $p(\tilde{f}_k \mid \tilde{e}_k)$ and $p(\tilde{e}_k \mid \tilde{f}_k)$ are the phrase lexicon models in both translation directions. The phrase translation probabilities are computed as a log-linear interpolation of the relative frequencies and the IBM-1 probability. The single word based lexicon models are denoted as $p(f_j \mid \tilde{e}_k)$ and $p(e_i \mid \tilde{f}_k)$, respectively. $p(f_j \mid \tilde{e}_k)$ is defined as the IBM-1 model probability of f_j over the whole phrase $\tilde{e}_k$, and $p(e_i \mid \tilde{f}_k)$ is the inverse model, respectively.

c_1 is the so-called word penalty, and c_2 is the phrase penalty, assigning constant costs to each target language word/phrase. The language model is a trigram model with modified Kneser-Ney discounting and interpolation (Stolcke, 2002). The search determines the target sentence and segmentation which maximize the objective function.

As equation 2 shows, the sub-models are combined via weighted log-linear interpolation. The model scaling factors $\lambda_1, \ldots, \lambda_5$ and the word and phrase penalties are optimized with respect to some evaluation criterion (Och, 2003), e.g. BLEU score.

3 Confidence measures for SMT

3.1 Related work

In this paper, we will present a new approach to word-level confidence estimation which makes explicit use of a phrase-based translation model. Most of the word-level confidence measures which have been presented in the literature so far are either based on relatively simple translation models such as IBM-1 (Blatz et al., 2003) or make use of information provided by the SMT system such as N-best lists or word graphs (Blatz et al., 2003; Gandrabur and Foster, 2003; Ueffing et al., 2003). In contrast to this, our method is based on a state-of-the-art statistical machine translation model, but nevertheless is independent of the machine translation system which generates the translation hypotheses.

The word-level confidence measures which showed the best performance in comparative experiments (Blatz et al., 2003) are word posterior probabilities and the IBM-1 based measure. Our new confidence measure will be compared to those approaches in section 7.3.

3.2 Word posterior probabilities

The confidence of a target word can be expressed by its posterior probability, i.e. the probability of the word to occur in the target sentence, given the source sentence. Consider a target word e occurring in the sentence in position i[1]. The posterior probability of this event can be determined by summing over all possible target sentences e_1^I containing the word e in position i:

$$p_i(e, f_1^J) \quad = \sum_{I, e_1^I:\, e_i = e} p(e_1^I, f_1^J) \qquad (3)$$

This value has to be normalized in order to obtain a probability distribution over all possible target words:

$$p_i(e \,|\, f_1^J) \quad = \quad \frac{p_i(e, f_1^J)}{\sum\limits_{e'} p_i(e', f_1^J)} \qquad (4)$$

[1]This is a rather strict assumption, because the position of a word in the target sentence can differ largely due to reorderings in the translation process. We present this variant here to keep the notation simple. Improved methods will be shown in the following sections.

4 System based confidence measures

In this section, we will present confidence measures which are based on N-best lists or word graphs generated by the SMT system. Those are representations of the space of the most likely translations of the source sentence.

The summation given in equation 3 is performed over all sentences which are contained in the N-best list or word graph. For a more detailed description, see (Ueffing et al., 2003).

4.1 Word graph based approach

The word posterior probability $p_i(e \,|\, f_1^J)$ can be calculated over a word graph using the forward-backward algorithm.

Let n', n be nodes in a word graph, and (n', n) the directed edge connecting them. The edge is annotated with a target word which we denote by $e(n', n)$ and the probability which this word contributes to the overall sentence probability, denoted by $p(n', n)$.

The forward probability $\Phi_i(n', n)$ of an edge is the probability of reaching this edge from the source of the graph, where the word $e(n', n)$ is the i-th word on the path. It can be obtained by summing the probabilities of all incoming paths of length $i-1$, which allows for recursive calculation. This leads to the following formula:

$$\Phi_i(n', n) \quad = \quad p(n', n) \cdot \sum_{n''} \Phi_{i-1}(n'', n') \; .$$

The backward probability expresses the probability of completing a sentence from the current edge, i.e. of reaching the sink of the graph. It can be determined recursively in descending order of i as follows:

$$\Psi_i(n', n) \quad = \quad p(n', n) \cdot \sum_{n^*} \Psi_{i+1}(n, n^*) \; .$$

Using the forward-backward algorithm, the word posterior probability of word e in position i is determined by combining the forward and backward probabilities of all edges which are annotated with e. This yields

$$p_i(e, f_1^J) = \sum_{(n', n):\, e(n', n) = e} \frac{\Phi_i(n', n) \cdot \Psi_i(n', n)}{p(n', n)} \; . \quad (5)$$

Note that (for computational reasons) the term $p(n', n)$ is included both in the forward and in the backward probability so that we have to divide the product by this term.

To obtain a posterior probability, a normalization, as shown in equation 4, has to be performed. The normalization term $\alpha := \sum\limits_{e'} p_i(e', f_1^J)$ corresponds to the probability mass contained in the word graph and can be calculated by summing the backward probabilities of all outgoing edges leaving the source s of the graph:

$$\alpha = \sum_{(s,n)} \Psi_1(s,n) \; .$$

As stated above, the position of word e in the target sentence can vary due to reorderings in the translation process. Therefore, we would like to relax the condition that e has to occur exactly in position i. This can be achieved by introducing a window of size t over the neighboring target positions and computing the sum of the word posterior probabilities over all positions $i-t, \ldots, i, \ldots, i+t$. In our experiments we found that a window over ± 3 positions yields the best performance.

4.2 N-best list based approach

N-best lists are an alternative representation of the space of translation hypotheses. They have the advantage that the Levenshtein alignment between a hypothesis and all sentences contained in the list can be performed easily. This makes it possible to consider not only target sentences, which contain the word e exactly in a position i (as given in equation 3), but to allow for some variation.

Let $\mathcal{L}(e_1^I, \tilde{e}_1^{\tilde{I}})$ be the Levenshtein alignment between sentences e_1^I and $\tilde{e}_1^{\tilde{I}}$. Then, $\mathcal{L}_i(e_1^I, \tilde{e}_1^{\tilde{I}})$ denotes the Levenshtein alignment of word e_i, i.e. the word in sentence $\tilde{e}_1^{\tilde{I}}$ which e_i is Levenshtein-aligned to.

The word posterior probability is then calculated by summing over all target sentences containing word e in a position which is Levenshtein-aligned to i:

$$p_i(e|f_1^J, I, e_1^I) = \frac{p_i(e, f_1^J, I, e_1^I)}{\sum\limits_{e'} p_i(e', f_1^J, I, e_1^I)} \; ,$$

where

$$p_i(e, f_1^J, I, e_1^I) = \sum_{\tilde{I}, \tilde{e}_1^{\tilde{I}}:\, \mathcal{L}_i(e_1^I, \tilde{e}_1^{\tilde{I}})=e} p(\tilde{e}_1^{\tilde{I}}, f_1^J) \; . \quad (6)$$

The confidence of word e then depends on the source sentence f_1^J as well as the target sentence e_1^I, because the whole target sentence is relevant for the Levenshtein alignment.

5 Phrase-based confidence measures

In contrast to the approaches presented in section 4, the phrase-based confidence measures do not not use the context information at the sentence level, but only at the phrase level. We want to determine a sort of marginal probability $Q(e, f_1^J)$. Therefore, we extract all source phrases f_j^{j+s} which occur in the given source sentence. For such source phrases, we find the possible translations e_i^{i+t} in the bilingual phrase lexicon. The confidence of target word e is then calculated by summing over all phrase pairs (f_j^{j+s}, e_i^{i+t}) where the target part e_i^{i+t} contains the word e.

Let $p(e_i^{i+t})$ be the language model score of the target phrase together with the word penalty c_1, i.e.

$$p(e_i^{i+t}) = \prod_{i'=i}^{i+t} c_1 \cdot p(e_{i'} \mid e_{i'-2}^{i'-1})^{\lambda_1} \; .$$

Analogously, define $p(f_j^{j+s}, e_i^{i+t})$ as the score of the phrase pair which consists of the phrase penalty and the phrase and word lexicon model scores (cf. section 2.2). Following equation 2, the (unnormalized) confidence is then determined as:

$$Q(e, f_1^J) = \sum_{j=1}^{J} \sum_{s=0}^{\min\{s_{\max}, J-j\}} \sum_{e_i^{i+t}:\, e \in e_i^{i+t}} p(e_i^{i+t}) \cdot p(f_j^{j+s}, e_i^{i+t}) \; , \quad (7)$$

where $s \leq s_{\max}$ and t are source and target phrase lengths, $s_{\max}$ being the maximal source phrase length.

In equation 7, the language model only determines the probability of the words within the target part of the phrase, and not across the phrase boundaries, because we consider only the single target phrases without context. Therefore, we assumed that the language model would not have much influence on the confidence estimation and also investigated a model without a language model. The same holds for word and phrase penalty: In the translation process they are useful for adjusting the length of the generated target hypothesis and for assigning more weight to longer phrases. Since this does not make much sense in our setting, we also investigated confidence estimation without word and phrase penalty.

Note that the value calculated in equation 7 is not normalized. In order to obtain a word posterior probability, we divide this value by the sum over the (unnormalized) confidence of all target words:

$$p_{phr}(e \mid f_1^J) = \frac{Q(e, f_1^J)}{\sum\limits_{e'} Q(e', f_1^J)} \; . \tag{8}$$

Unlike the word posterior probabilities presented in the previous section, this value is completely independent of the target sentence position in which the word e occurs.

As stated in section 2.2, the scaling factors of the different sub-models and the penalties in the translation system are optimized with respect to some evaluation criterion. But since the values which are optimal for translation are not necessarily optimal for confidence estimation, we perform optimization here as well: We train the probability models on the training corpus, estimate the word confidences on the development corpus, and optimize the scaling factors with respect to the classification error rate described in section 7.2. The optimization is performed with the Downhill Simplex algorithm (Press et al., 2002).

6 IBM-1 based approach

Another type of confidence measure which does not rely on system output and is thus applicable to any kind of machine translation system is the IBM-1 model based confidence measure which was introduced in (Blatz et al., 2003). We modified this confidence measure because we found that the average lexicon probability used there is dominated by the maximum. Therefore, we determine the *maximal* translation probability of the target word e over the source sentence words:

$$p_{\text{IBM}-1}(e|f_1^J) = \max_{j=0,\dots,J} p(e|f_j) \; , \tag{9}$$

where f_0 is the "empty" source word (Brown et al., 1993). The probabilities $p(e|f_j)$ are word-based lexicon probabilities.

Investigations on the use of the IBM-1 model for word confidence measures showed promising results (Blatz et al., 2003; Blatz et al., 2004). Thus, we apply this method here in order to compare it to the other types of confidence measures.

7 Experiments

7.1 Experimental setting

The experiments were performed on three different language pairs. All corpora were compiled in the EU project TransType2; they consist of technical manuals. The corpus statistics are given in table 1. The SMT systems that the confidence estimation was performed for were trained on these corpora. The same holds for the probability models that were used to estimate the word confidences.

We used several (S)MT systems for testing the confidence measures. A detailed analysis will be given for two of them; the so-called alignment template system (Och and Ney, 2004), (denoted as AT in the tables) and the phrase-based translation system described in section 2.2 (denoted as PBT in the tables). They are both state-of-the-art SMT systems. We produced single best translations, word graphs and N-best lists on all three language pairs using these systems. The translation quality in terms of WER, PER (position independent word error rate), BLEU and NIST score is given in tables 2 and 3. We see that the best results are obtained on Spanish to English translation, followed by French to English and German to English.

Two more translation systems were used for comparative experiments: One is a statistical MT system which is based on a finite state architecture (FSA). For a description of this system, see (Kanthak et al., 2005). Additionally, we used translations generated by Systran[2]. Table 3 presents the translation error rates and scores for all systems on the German $\rightarrow$ English test corpus. These hypotheses were used to investigate whether the phrase-based confidence measures perform well independently of the translation system.

All three SMT systems (AT, PBT and FSA) show very similar performance on the German $\rightarrow$ English test corpus. The fact that Systran generates translations of much lower quality is due to the fact that the technical manuals are very specific in terminology, and the SMT systems have been trained on similar corpora.

[2]http://babelfish.altavista.com/tr, June 2005

Table 1: Statistics of the training, development and test corpora.

		French	English	Spanish	English	German	English
TRAIN	Sentences	53 046		55 761		49 376	
	Running Words	680 796	628 329	752 606	665 399	537 464	589 531
	Vocabulary	15 632	13 816	11 050	7 956	23 845	13 223
DEV	Sentences	994		1 012		964	
	Running Words	11 674	10 903	15 957	14 278	10 462	10 642
	OOVs	184	141	54	27	147	29
TEST	Sentences	984		1 125		996	
	Running Words	11 709	11 177	10 106	8 370	11 704	12 298
	OOVs	204	201	69	49	485	141

Table 2: Translation quality of systems AT and PBT on the test corpora described in table 1.

	AT		PBT	
	S→E	F→E	S→E	F→E
WER[%]	29.6	54.8	26.1	54.9
PER[%]	20.1	43.7	17.5	43.4
BLEU[%]	63.4	31.5	66.9	31.3
NIST	8.80	6.64	8.98	6.62

Table 3: Translation quality of all MT systems on the German → English test corpus.

	AT	PBT	FSA	Systran
WER[%]	62.7	61.6	63.2	79.2
PER[%]	49.8	49.6	50.4	66.4
BLEU[%]	26.6	25.7	26.5	12.0
NIST	5.92	5.72	5.79	4.09

To determine the true class of each word in a generated translation hypothesis, we use the word error rate (WER). That is, a target word is considered correct if it is aligned to itself in the Levenshtein alignment between hypothesis and reference translation(s). We also investigated PER based classification, but since the tendencies of the results were similar, we omit them here.

7.2 Evaluation metrics

After computing the confidence measure, each generated word is tagged as either *correct* or *false*, depending on whether its confidence exceeds the tagging threshold that has been optimized on the development set beforehand. The performance of the confidence measure is evaluated using the **C**lassification **E**rror **R**ate (CER). This is defined as the number of incorrect tags divided by the total number of generated words in the translated sentence. The baseline CER is determined by assigning the most frequent class to all translations. In the case that the most frequent class is "correct" (meaning at least half of the words in the generated translation are correct w.r.t. to WER), this is the number of substitutions and insertions, divided by the number of generated words. The CER strongly depends on the tagging threshold. Therefore, the tagging threshold is adjusted beforehand (to minimize CER) on a development corpus *distinct* to the test set.

7.3 Experimental results

Table 4 shows the performance of all different confidence measures on the hypotheses generated by the alignment template system and the phrase-based system. For the baseline CER, we determined the 90%- and 99%-confidence intervals using the bootstrap estimation method described in (Bisani and Ney, 2004)[3]. We see that, in all settings but one, the word graph and the N-best list based method outperform the IBM-1 based confidence measure. On French → English, the improvement over the baseline is significant at the 1%-level for these methods, whereas on Spanish → English this is only the case at 10%. The performance of the N-best list based approach is better than that of the word graph based

[3]The tool is freely available from http://www-i6.informatik.rwth-aachen.de/web/Software/index.html

confidence measures for the alignment template system. This is probably due to the fact that the former can take the Levenshtein alignment into account and thus estimate the word confidence more reliably.

The phrase-based confidence measures show a performance which is clearly better than that of the other methods. We obtain a relative improvement of up to 7.8% over the best existing method on these language pairs. The improvement over the baseline is significant even at the 1%-level in all cases.

When analyzing the impact of the different submodels in the phrase-based approach, we found that the language model does not have much impact on the confidence estimation. There are only slight variations in the CER if the model is omitted. The word and phrase penalty on the other hand seem to be important (with one exception in the first setting).

The evaluation of the system-independent confidence measures (i.e. those based on IBM-1 and the new phrase-based method we presented) for four different translation systems is shown in table 5. We see that, for all of them, the phrase-based approach outperforms the IBM-1 based method significantly. The largest gain in terms of CER is achieved for the Systran translations: 23.8% relative over the IBM-1 based measure.

8 Conclusion and outlook

We presented a new approach to word-level confidence estimation for machine translation which makes use of bilingual phrases. By using models from a state-of-the-art phrase-based statistical machine translation system, the word confidences are estimated only on the basis of single best system output. Unlike other confidence measures, this does not rely on information from the machine translation system which generated the translation.

Experimental evaluation on three different language pairs and on output from structurally different translation systems showed that the new confidence measures perform better than existing confidence measures in all cases. The application on output from different MT systems yielded a significant reduction of the error rate over the existing measures. This proves that the method is well-suited for word confidence estimation on statistical as well as non-statistical MT systems.

The task investigated in this work was a text translation task in the domain of technical manuals. We are currently investigating the use of word-level confidence measures on data from the European parliament. It will be interesting to see whether a similar performance can be achieved on this large vocabulary speech translation task.

Acknowledgement

This work was partly funded by the European Union under the RTD project TransType2 (IST–2001–32091), and under the integrated project TC-STAR – Technology and Corpora for Speech to Speech Translation (IST-2002-FP6-506738).

References

Y. Akiba, E. Sumita, H. Nakaiwa, S. Yamamoto, and H. G. Okuno. 2004. Using a mixture of n-best lists from multiple MT systems in rank-sum-based confidence measure for MT outputs. In *Proc. CoLing*, pages 322–328, August.

N. Bertoldi, R. Cattoni, M. Cettolo, and M. Federico. 2004. The ITC-irst statistical machine translation system for IWSLT-2004. In *Proc. IWSLT*, pages 51–58, Kyoto, Japan, September.

M. Bisani and H. Ney. 2004. Bootstrap estimates for confidence intervals in asr performance evaluationx. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 409–412, Montreal, Canada, May.

J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. 2003. Confidence estimation for machine translation. Final report, JHU/CLSP Summer Workshop. `http://www.clsp.jhu.edu/ws2003/groups/estimate/`.

J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. 2004. Confidence estimation for machine translation. In *Proc. CoLing*, pages 315–321, Geneva, Switzerland, August.

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, June.

S. Gandrabur and G. Foster. 2003. Confidence estimation for text prediction. In *Proc. CoNLL*, pages 95–102, Edmonton, Canada, May.

S. Jayaraman and A. Lavie. 2005. Multi-engine machine translation guided by explicit word matching.

Table 4: CER for different confidence measures, reference based on WER. Hypotheses from the alignment template system and the phrase-based system. The best value is printed in bold.

	alignment template system		phrase-based system	
Model	S → E	F → E	S → E	F → E
Baseline	20.8	42.5	19.2	42.7
99%-confidence interval	[18.8,22.7]	[40.1,44.7]	[17.2,21.2]	[40.4,45.0]
90%-confidence interval	[19.6,22.1]	[40.9,43.9]	[17.9,20.5]	[41.2,44.2]
Word graphs from the system (eq. 5)	20.1	32.9	17.9	30.5
N-best lists from the system (eq. 6)	19.8	31.9	17.9	30.9
phrase-based (eq. 8)	17.5	**30.2**	**16.5**	**30.0**
without language model	**17.4**	30.3	**16.5**	30.3
without word and phrase penalty	17.5	30.6	16.9	30.5
IBM-1 (eq. 9)	20.0	34.1	18.3	35.1

Table 5: CER for different confidence measures on the German → English test set, reference based on WER. Hypotheses from different MT systems.

Model	AT	PBT	FSA	Systran
Baseline	49.2	48.4	46.6	37.4
99%-confidence interval	[48.6,53.1]	[45.9,50.7]	[44.2,49.0]	[36.0,38.9]
phrase-based (eq. 8)	27.6	26.4	30.2	24.3
IBM-1 (eq. 9)	32.8	32.8	37.0	31.9

In *Proc. EAMT*, pages 143–152, Budapest, Hungary, May.

S. Kanthak, D. Vilar, E. Matusov, R. Zens, and H. Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *ACL 2005 – Proc. Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 167–174, Ann Arbor, Michigan, June.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL*, pages 127–133, Edmonton, Canada, May/June.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, December.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167, Sapporo, Japan, July.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2002. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK.

C. Quirk. 2004. Training a sentence-level machine translation confidence metric. In *Proc. LREC*, pages 825–828, Lisbon, Portugal, May.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. ICSLP*, volume 2, pages 901–904, Denver.

C. Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proc. EMNLP*, pages 1–8, Sapporo, Japan, July.

N. Ueffing and H. Ney. 2005. Application of word-level confidence measures in interactive statistical machine translation. In *Proc. EAMT*, pages 262–270, Budapest, Hungary, May.

N. Ueffing, K. Macherey, and H. Ney. 2003. Confidence Measures for Statistical Machine Translation. In *Proc. MT Summit IX*, pages 394–401, New Orleans, LA, September.

S. Vogel, S. Hewavitharana, M. Kolss, and A. Waibel. 2004. The ISL statistical translation system for spoken language translation. In *Proc. IWSLT*, pages 65–72, Kyoto, Japan, September.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proc. HLT-NAACL*, pages 257–264, Boston, MA, May.

Word-Sense Disambiguation for Machine Translation

David Vickrey **Luke Biewald** **Marc Teyssier** **Daphne Koller**
Department of Computer Science
Stanford University
Stanford, CA 94305-9010
{dvickrey,lukeb,teyssier,koller}@cs.stanford.edu

Abstract

In word sense disambiguation, a system attempts to determine the sense of a word from contextual features. Major barriers to building a high-performing word sense disambiguation system include the difficulty of labeling data for this task and of predicting fine-grained sense distinctions. These issues stem partly from the fact that the task is being treated in isolation from possible uses of automatically disambiguated data. In this paper, we consider the related task of word translation, where we wish to determine the correct translation of a word from context. We can use parallel language corpora as a large supply of partially labeled data for this task. We present algorithms for solving the word translation problem and demonstrate a significant improvement over a baseline system. We then show that the word-translation system can be used to improve performance on a simplified machine-translation task and can effectively and accurately prune the set of candidate translations for a word.

1 Introduction

The problem of distinguishing between multiple possible senses of a word is an important subtask in many NLP applications. However, despite its conceptual simplicity, and its obvious formulation as a standard classification problem, achieving high levels of performance on this task has been a remarkably elusive goal.

In its standard formulation, the disambiguation task is specified via an ontology defining the different senses of ambiguous words. In the Senseval competition, for example, WordNet (Fellbaum, 1998) is used to define this ontology. However, ontologies such as WordNet are not ideally suited to the task of word-sense disambiguation. In many cases, WordNet is overly "specific", defining senses which are very similar and hard to distinguish. For example, there are seven definitions of "respect" *as a noun* (including closely related senses such as "an attitude of admiration or esteem" and "a feeling of friendship and esteem"); there are even more when the verb definitions are included as well. Such closely related senses pose a challenge both for automatic disambiguation and hand labeling. Moreover, the use of a very fine-grained set of senses, most of which are quite rare in practice, makes it very difficult to obtain sufficient amounts of training data.

These issues are clearly reflected in the performance of current word-sense disambiguation systems. When given a large amount of training data for a particular word with reasonably clear sense distinctions, existing systems perform fairly well. However, for the "all-words" task, where all ambiguous words from a test corpus must be disambiguated, it has so far proved difficult to perform significantly better than the baseline heuristic of choosing the most common sense for each word.[1]

In this paper, we address a different formulation of the word-sense disambiguation task. Rather than considering this task on its own, we consider a task of disambiguating words for the purpose of some larger goal. Perhaps the most direct and compelling application of a word-sense disambiguator is to machine translation. If we knew the correct semantic meaning of each word in the source language, we could more accurately determine the appropriate words in the target language. Importantly, for this application, subtle shades of meaning will often be irrelevant in choosing the most appropriate words in the target language, as closely related senses of a single word in one language are often encoded by a single word in another. In the context of this larger goal, we can focus only on sense distinctions that a human would consider when choosing the translation of a word in the source language.

We therefore consider the task of word-sense disambiguation for the purpose of machine translation. Rather than predicting the sense of a particular word a, we predict the possible translations of a into the

[1]See, for example, results of Senseval-3, available at http://www.senseval.org/senseval3

target language. We both train and evaluate the system on this task. This formulation of the word-sense disambiguation task, which we refer to as *word translation*, has multiple advantages. First, a very large amount of "partially-labeled" data is available for this task in the form of bilingual corpora (which exist for a wide range of languages). Second, the "labeling" of these corpora (that is, translation from one language to another), is a task at which humans are quite proficient and which does not generally require the labeler (translator) to make difficult distinctions between fine shades of meaning.

In the remainder of this paper, we first discuss how training data for this task can be acquired automatically from bilingual corpora. We apply a standard learning algorithm for word-sense disambiguation to the word translation task, with several modifications which proved useful for this task.We present the results of our algorithm on word translation, showing that it significantly improves performance on this task. We also consider two simple methods for incorporating word translation into machine translation. First, we can use the output of our model to help a translation model choose better words; since general translation is a very noisy process, we present results on a simplified translation task. Second, we show that the output of our model can be used to prune candidate word sets for translation; this could be used to significantly speed up current translation systems.

2 Machine Translation

In machine translation, we wish to translate a sentence $\mathbf{s}$ in our source language into $\mathbf{t}$ in our target language. The standard approach to statistical machine translation uses the *source-channel model* ,

$$\text{argmax}_{\mathbf{t}} P(\mathbf{t}|\mathbf{s}) = \text{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{s}|\mathbf{t}),$$

where $P(\mathbf{t})$ is the *language model* for the target language, and $P(\mathbf{s}|\mathbf{t})$ is an *alignment model* from the target language to the source language. Together they define a generative model for the source/target pair $(\mathbf{s}, \mathbf{t})$: first $\mathbf{t}$ is generated according to the language model $P(\mathbf{t})$; then $\mathbf{s}$ is generated from $\mathbf{t}$ according to $P(\mathbf{s}|\mathbf{t})$.[2]

Typically, strong independence assumptions are then made about the distribution $P(\mathbf{s}|\mathbf{t})$. For example, in the IBM Models (Brown et al., 1993), each word t_i independently generates 0, 1, or more words in the source language. Thus, the words generated by t_i are independent of the words generated by t_j for each $j \neq i$. This means that correlations between words in the source sentence are not captured by $P(\mathbf{s}|\mathbf{t})$, and so the context we will use in our word translation models to predict t_i given s_i is not available to a system making these independence assumptions. In this type of system, semantic and syntactic relationships between words are only modeled in the target language; most or all of the semantic and syntactic information contained in the source sentence is ignored. The language model $P(\mathbf{t})$ does introduce some context-dependencies, but the standard n-gram model used in machine translation is too weak to provide a reasonable solution to the strong independence assumptions made by the alignment model.

3 Task Formulation

We define the word translation task as finding, for an individual word a in the source language $\mathcal{S}$, the correct translation, either a word or phrase, in the target language $\mathcal{T}$. Clearly, there are cases where a is part of a multi-word phrase that needs to be translated as a unit. Our approach could be extended by preprocessing the data in $\mathcal{S}$ to find phrases, and then executing the entire algorithm treating phrases as atomic units. We do not explore this extension in this paper, instead focusing on the word-to-phrase translation problem.

As we discussed, a key advantage of the word translation vs. word sense disambiguation is the availability of large amounts of training data. This data is in the form of bilingual corpora, such as the European Parliament proceedings[3]. Such documents provide many training instances, where a word in one language is translated into another. However, the data is only partially labeled in that we are not given a word-to-word alignment between the two languages, and thus we do not know what every word in the source language $\mathcal{S}$ translates to in the target language $\mathcal{T}$. While sentence-to-sentence alignment is a fairly easy task, word-to-word alignment is considerably more difficult. To obtain word-to-word alignments, we used GIZA++[4], an implementation of the IBM Models (specifically, we used the output of IBM Model 4). We did not perform stemming on either language, so as to preserve suffix information for our word translation system and the machine translation language model.

Let $D_{\mathcal{S}}$ be the set of sentences in the source lan-

[2]Note that we refer to $\mathbf{t}$ as the target sentence, even though in the source-channel model, $\mathbf{t}$ is the source sentence which goes through the channel model $P(\mathbf{s}|\mathbf{t})$ to produce the observed sentence $\mathbf{s}$.

[3]Available at http://www.isi.edu/ koehn/

[4]Available at http://www.isi.edu/ och/GIZA++.html

French (frequency)	Translation
montée(51)	going up
lève(10), lever(17)	standing up
hausse(58), augmenter(37), augmentation(150)	increase(number)
interviens(53)	to rise to speak
naissance(21), source(10)	to be created, arise
soulevé(10)	raising an issue

Table 1: Aligned translations for "rise" occurring at least 10 times in the corpus

guage and $D_{\mathcal{T}}$ the set of target language sentences.

The alignment algorithm can be run in either direction. When run in the $\mathcal{S} \rightarrow \mathcal{T}$ direction, the algorithm aligns each word in $\mathbf{t}$ to at most one word in $\mathbf{s}$. Consider some source sentence $\mathbf{s}$ that contains the word a, and let $U_{a,\mathbf{s}\rightarrow\mathbf{t}} = b_1, \ldots, b_k$ be the set of words that align to a in the aligned sentence $\mathbf{t}$. In general, we can consider $U_a = \{U_{a,\mathbf{s}\rightarrow\mathbf{t}}\}_{\mathbf{s}\in D_a}$ to be the candidate set of translations for a in $\mathcal{T}$, where D_a is the set of source language sentences containing a. However, this definition is quite noisy: a word b_i might have been aligned with a arbitrarily; or, b_i might be a word that itself corresponds to a multi-word translation in $\mathcal{S}$. Thus, we also align the sentences in the $\mathcal{T} \rightarrow \mathcal{S}$ direction, and require that each b_i in the phrase aligns either with a or with nothing. As this process is still fairly noisy, we only consider a word or phrase $b \in U_a$ to be a candidate translation for a if it occurs some minimum number of times in the data.

For example, Table 1 shows a possible candidate set for the English word "rise", with French as the target language. Note that this set can contain not only target words corresponding to different meanings of "rise" (the rows in the table) but also words which correspond to different grammatical forms in the target language corresponding to different parts of speech, verb tenses, etc. So, disambiguation in this case is both over senses and grammatical forms.

The final result of our processing of the corpus is, for each source word a, a set of target words/phrases U_a; and a set of sentences D_a where, in each sentence, a is aligned to some $b \in U_a$. For any sentence $\mathbf{s} \in D_a$, aligned to some target sentence $\mathbf{t}$, let $u_{a,\mathbf{s}} \in U_a$ be the word or phrase in $\mathbf{t}$ aligned with a. We can now treat this set of sentences as a fully-labeled corpus, which can be split into a set used for learning the word-translation model and a test set used for evaluating its performance.

We note, however, that there is a limitation to using accuracy on the test set for evaluating the performance of the algorithm. A source word a in a given context may have two equally good, interchangeable translations into the target language. Our evaluation metric only rewards the algorithm for selecting the target word/phrase that happened to be used in the actual translation. Thus, accuracies measured using this metric may be artificially low. This is a common problem with evaluating machine translation systems.

Another issue is that we take as ground truth the alignments produced by GIZA++. This has two implications: first, our training data may be noisy since some alignments may be incorrect; and second, our test data may not be completely accurate. As mentioned above, we only consider possible translations which occur some minimum number of times; this removes many of the mistakes made by GIZA++. Even if the test set is not 100% reliable, though, improvement over baseline performance is indicative of the potential of a method.

4 Word Translation Algorithms

The word translation task and the word-sense disambiguation task have the same form: each word a is associated with a set of possible labels U_a; given a sentence $\mathbf{s}$ containing word a, we must determine which of the possible labels in U_a to assign to a in the context $\mathbf{s}$. The only difference in the two tasks is the set U_a: for word translation it is the set of possible translations of a, while for word sense disambiguation it is the set of possible senses of a in some ontology. Thus, we may use any word sense disambiguation algorithm as a word translation algorithm by appropriately defining the senses (assuming that the WSD algorithm does not assume that a particular ontology is used to choose the senses).

Our main focus in this paper is to show that machine learning techniques are effective for the word translation task, and to demonstrate that we can use the output of our word translation system to improve performance on two machine-translation related tasks. We will therefore restrict our attention to a relatively simple model, logistic regression (Minka, 2000). There are several motivations for using this discriminative, probabilistic model. First, it is known both theoretically and empirically (e.g., (Ng and Jordan, 2002)) that discriminative models achieve higher accuracies than generative models if enough data is available. For the traditional word-sense disambiguation task, data must be hand-labeled, and is therefore often too scarce to allow for discriminative training. In our setting, however, training data is acquired automatically from bilingual corpora, which are widely available and quite large. Thus, discriminative training is a viable option for the word translation problem. A second consideration is that, to effectively incorporate our system into a statistical machine translation system, we would like to produce not just a single prediction, but a list of confidence-rated possibilities. The optimization procedure of logistic regression attempts to produce a distribution over possible translations which accurately represents the confidence of the model for each translation. By contrast, a classical Naive Bayes model often assigns very low probabilities to all but the most likely translation. Other word-sense disambiguation models may not produce confidence measures at all.

Features. Our word translation model for a word a in a sentence $\mathbf{s} = w_1, \ldots, w_k$ is based on features constructed from the word and its context within the sentence. Our basic logistic regression model uses the following features, which correspond to the feature space for a standard Naive Bayes model:

- the part of speech of a (generated using the Brill tagger)[5];
- a binary "occurs" variable for each word which is 1 if that word is in a fixed context centered at a (c_r words to the right and c_l words to the left), and 0 otherwise.

We also consider an extension to this model, where instead of the fixed context features above, we use:

- for each direction $d \in \{l, r\}$ and each possible context size $c_d \in \{1, ..., C_d\}$, an "occurs" variable for each word.

This is a true generalization of the previous context features, since it contains features for all possible context sizes, not just one particular fixed size. This feature set is equivalent to having one feature for each word in each context position, except that it will have a different prior over parameters under standard L_2 regularization. This feature set allows our model to distinguish between very local (often syntactic) features and somewhat longer range features whose exact position is not as important.

Let $\phi^{a,\mathbf{s}}$ be the set of features for word a to be translated, with sentence context $\mathbf{s}$ (the description of the model does not depend on the particular feature set selected).

Model. The logistic regression model encodes the conditional distribution $(P(u_{a,\mathbf{s}} = b \mid a, \mathbf{s}) : b \in U_a)$. Such a model is parameterized by a set of vectors $\boldsymbol{\theta}^a_b$, one for each word a and each possible target $b \in U_a$, where each vector contains a weight $\theta^a_{b,j}$ for each feature $\phi^{a,\mathbf{s}}_j$. We can now define our conditional distribution:

$$P_{\boldsymbol{\theta}^a}(b \mid a, \mathbf{s}) = \frac{1}{Z_{a,\mathbf{s}}} e^{\boldsymbol{\theta}^a_b \phi^{a,\mathbf{s}}}$$

with partition function $Z_{a,\mathbf{s}} = \sum_{b' \in U_a} \exp(\boldsymbol{\theta}^a_{b'} \phi^{a,\mathbf{s}})$.

Training. We train the logistic regression model to maximize the conditional likelihood of the observed labels given the features in our training set. Thus, our goal in training the model for a is to maximize

$$\prod_{\mathbf{s} \in D_a} P_{\boldsymbol{\theta}^a}(u_{a,\mathbf{s}} \mid a, \mathbf{s}).$$

We maximize this objective by maximizing its logarithm (the log-conditional-likelihood) using conjugate gradient ascent (Shewchuk, 1994).

One important consideration when training using maximum likelihood is regularization of the parameters. In the case of logistic regression, the most common type of regularization is L_2 regularization; we then maximize

$$\prod_{b,j} \exp\left(-\frac{(\theta^a_{b,j})^2}{2\sigma^2}\right) \prod_{\mathbf{s} \in D_a} P_{\boldsymbol{\theta}^a}(u_{a,\mathbf{s}} \mid a, \mathbf{s}).$$

This penalizes the likelihood for the distance of each parameter $\theta^a_{b,j}$ from 0; it corresponds to a Gaussian prior on each parameter with variance σ^2.

5 Word Translation Results

For our word translation experiments we used the European Parliament proceedings corpus, which contains approximately 27 million words in each of English and French (as well as a number of other languages). We tested on a set of 1859 ambiguous words — specifically, all ambiguous words contained in the first document of the corpus. For each of these words, we found all instances of the word in the corpus and split these instances into training and test sets.

We tested four different models. The first, Baseline, always chooses the most common translation for the word; the second, Baseline with Part of Speech, uses tagger-generated parts of speech to choose the most common translation for the observed word/part-of-speech pair. The third model, Simple Logistic, is the logistic regression model with the simpler feature set, a context window of a fixed size. We selected the window size by evaluating accuracy for a variety of window sizes on 20 of the 1859 ambiguous words using a random train-test split. The window size which performed best on average extended one word to the left and

[5] Available at http://www.cs.jhu.edu/ brill/

Model	Macro	Micro
Baseline	0.511	0.526
Baseline with Part of Speech	0.519	0.532
Simple Logistic	0.581	0.605
Logistic	0.596	0.620

Table 2: Average Word Translation Accuracy

two words to the right (larger windows generally resulted in overfitting). The fourth model, Logistic, is the logistic regression model with overlapping context windows; the maximum window size for this model was four words to the left and four words to the right. We selected the standard deviation σ^2 for the logistic models by trying different values on the same small subset of the ambiguous words. For the Simple Logistic model, the best value was $\sigma^2 = 1$; for the Logistic model, it was 0.35.

Table 2 shows results of these four models. The first column is macro-averaged over the 1859 words, that is, the accuracy for each word counts equally towards the average. The second column shows the micro-averaged accuracy, where each test example counts equally. We will focus on the micro-averaged results, since they correspond to overall accuracy.

The less accurate of our two models, Simple Logistic, improves around 8% over the simple baseline and 7% over the part-of-speech baseline on average. Our more complex logistic model, which is able to handle larger context sizes without significantly overfitting, improves accuracy by another 1.5%.

There was a great deal of variance from word to word in the performance of our models relative to baseline. For a few words, we achieved very large increases in accuracy. For instance, the noun "agenda" showed a 31.2% increase over both baselines. Similarly, the word "rise" (either a noun or a verb) had part-of-speech baseline accuracy of 27.9%. Our model increased the accuracy to 57.0%.

It is worth repeating that accuracies on this task are artificially low since in many cases a single word can be translated to many different words with the same meaning. At the same time, accuracies are artificially inflated by the fact that we only consider examples where we can find an aligned word in the French corpus, so translations where a word is dropped or translated as part of a compound word are not counted.

One disadvantage of the EuroParl corpus is that it is not "balanced" in terms of semantic content. It is not clear how this affects our results.

6 Blank-Filling Task

One of the most difficult parts of machine translation is decoding — finding the most likely translation according to some probability model. The difficulty arises from the enormous number of possible translated sentences. Existing decoders generally use either highly pruned search or greedy heuristic search. In either case, the quality of a translation can vary greatly from sentence to sentence. This variation is much higher than the improvement in "semantic" accuracy our model is attempting to achieve. Moreover, currently available decoders do not provide a natural way to incorporate the results of a word translation system. For example, Carpuat and Wu (2005) obtain negative results for two methods of incorporating the output of a word-sense disambiguation system into a machine translation system.

Thus, we instead used our word translation model for a simplified translation problem. We prepared a dataset as follows: for each occurrence of an ambiguous words in an English sentence in the first document of the Europarl corpus, we tried to determine what the correct translation for that word was in the corresponding French sentence. If we found one and exactly one possible translation for that word in the French sentence, we replaced that word with a "blank", and linked the English word to that blank. The final result was a set of 655 sentences with a total of 3018 blanks.

For example, the following English-French sentence pair contains the two ambiguous words *address* and *issue* and one possible translation for each, *examiner* and *question*:

- Therefore, the commission should *address* the *issue* once and for all.
- Par conséquent, la commission devra enfin *examiner* cette *question* particulière.

We replace the translations of the ambiguous words with blanks; we would like a decoder to replace the blanks with the correct translations:

- Par conséquent, la commission devra enfin [*address*] cette [*issue*] particulière.

An advantage of this task is that, for a given distribution $P(\mathbf{t}|\mathbf{s})$, we can easily write a decoder which exhaustively searches the entire solution space for the best answer (provided that there are not too many blanks and that $P(\mathbf{t}|\mathbf{s})$ is sufficiently "local" with respect to $\mathbf{t}$). Thus, we can be sure that it is the probability model, and not the decoder, which is determining the quality of the output. Also, we have removed most or all syntactic variability from the task,

Model	λ_{lm}	λ_{ga}	λ_{da}	λ_{wt}	Acc
Language Model only	1	0	0	0	0.749
Source-Channel	1	1	0	0	0.821
LM + GA + DA	1	0.6^*	0.6^*	0	0.833
LM + GA + DA + WT	1	0.6^*	0^*	1.2^*	0.846

Table 3: Blank-filling results. Weights marked with * have been optimized.

allowing us to better gauge whether we are choosing *semantically* correct translations.

Let (a_i, b_i) be the pairs of words corresponding to the blanks in sentence $\mathbf{t}$. Then the alignment model decomposes as a product of terms over these pairs, e.g. $P(\mathbf{s}|\mathbf{t}) \propto \prod_{(a_i,b_i)} P(a_i|b_i)$. Analogously, we extend the word translation model as $P_{wt}(\mathbf{t}|\mathbf{s}) \propto \prod_{(a_i,b_i)} P_{wt}(b_i|\mathbf{s}, a_i)$.

The source-channel model can be used directly to solve the blank filling task; the language model makes use of the French words surrounding each blank, while the alignment model guesses the appropriate translation based on the aligned English word. As we have mentioned, this model does not take full advantage of the context in the English sentence. Thus, we hope that incorporating the word translation model into the decoder will improve performance on this task.

Conversely, simply using the word translation model alone for the blank-filling task would not take advantage of the available French context. There are four probability distributions we might consider using: the language model $P_{lm}(\mathbf{t})$; the "generative" alignment model $P_{ga}(\mathbf{s}|\mathbf{t})$, which we calculate using the training samples from the previous section; the analogous "discriminative" alignment model $P_{da}(\mathbf{t}|\mathbf{s})$, which corresponds to the Baseline system we compared to on the word translation task; and our overlapping context logistic model, $P_{wt}(\mathbf{t}|\mathbf{s})$, which also goes in the "discriminative" direction, but uses the context features in the source language for determining the distribution over each word's possible translations.

We combine these models by simply taking a log-linear combination:

$$\log P(\mathbf{t}|\mathbf{s}) \propto \lambda_{lm} \log P_{lm}(\mathbf{t}) + \lambda_{ga} \log P_{ga}(\mathbf{s}|\mathbf{t}) + \lambda_{da} \log P_{da}(\mathbf{t}|\mathbf{s}) + \lambda_{wt} \log P_{wt}(\mathbf{t}|\mathbf{s}).$$

The case of $\lambda_{lm} = \lambda_{ga} = 1$ and $\lambda_{da} = \lambda_{wt} = 0$ reduces to the source-channel model; other settings incorporate discriminative models to varying degrees.

We evaluated this combined translation model on the blank-filling task for various settings of the mixture coefficients λ. For our language model we used the CMU-Cambridge toolkit.[6] The word translation model for each ambiguous word was trained on all documents except the first.

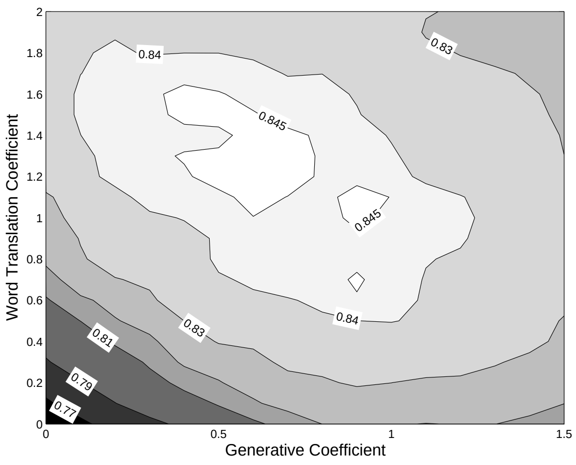


Figure 1: Accuracy on blank-filling task with $\lambda_{lm} = 1$ and $\lambda_{disc} = 0$ as a function of λ_{gen} and λ_{wt}.

Table 3 shows results for several sets of weights. A * denotes entries which we optimized (see below); other entries were fixed. For example, the third model was obtained by fixing the coefficient of the language model to 1 and the word-translation to 0, and optimizing the weights for the generative and discriminative alignment models.

The language model alone is able to achieve reasonable results; adding the alignment models improves performance further. By adding the word-translation model, we are able to improve performance by approximately 2.5% over the source-channel model, a relative error reduction of 14%, and 1.3% over the optimized model using the language model and generative and discriminative alignment models, a relative error reduction of 7.8%.

We chose optimal coefficients for the combined probability models by exhaustively trying all possible settings of the weights, at a resolution of 0.1, evaluating accuracy for each one on the test set. Figure 1 shows the performance on the blank-filling task as a function of the weights of the generative alignment model and the word-translation model (the optimum value of the discriminative alignment model $P(\mathbf{t}|\mathbf{s})$ is always 0 when we include the word-translation model). As we can see, the performance of this model is robust with respect to the exact value of the coefficients. The "obvious" setting of 1.0 for the generative model and 1.0 for the word translation model performs nearly as well

[6]Available at http://mi.eng.cam.ac.uk/ prc14/toolkit.html.

as the optimized setting. In the optimal region, the word-translation model receives twice as much weight as the generative alignment model, indicating that word-translation model is more informative than the generative alignment model. Incorporating the discriminative alignment model into the source-channel model also improves performance, but not nearly as much as using the word-translation model.

An alternate way to optimize weights over translation features is described in Och and Ney (2002). They consider a number of translation features, including the language model and generative and discriminative alignment models.

7 Search Space Pruning

As we have mentioned, one of the main difficulties in translation is that there are an enormous number of possible translations to consider. Decoding algorithms must therefore use some kind of search-space pruning in order to be efficient. A key part of pruning the search space is deciding on the set of words to consider in possible translations (Germann et al., 2001). One standard method is to consider only target words which have high probability according to the discriminative alignment model. But we have already shown that the word translation model achieves much better performance on word translation than this baseline model; thus, we would expect the word translation model to improve accuracy when used to pick sets of candidate translations.

Given a probability distribution over possible translations of a word, $P(b|a, \mathbf{s})$, there are several ways to choose a reduced set of possible translations. Two commonly used methods are to only consider the top n scoring words from this distribution (*best-n*); and to only consider words b such that $P(b|a, \mathbf{s})$ is above some fixed threshold (*cut-off*).

We use the same data set as for the blank-filling task. We evaluate the accuracy of a pruning strategy by evaluating whether the correct translation is in the candidate set selected by the pruning strategy. To compare results for different pruning strategies, we plot performance as a function of average size of the candidate translation set. Figure 2 shows the accuracy vs. average candidate set size for the word-translation model, discriminative alignment model, and generative alignment model.

The generative alignment model has the worst performance of the three. This is not surprising as it does not take into account the prior probability of the target word $P(b)$. More interestingly, we see that the word-translation model outperforms the discriminative translation model by a significant amount. For

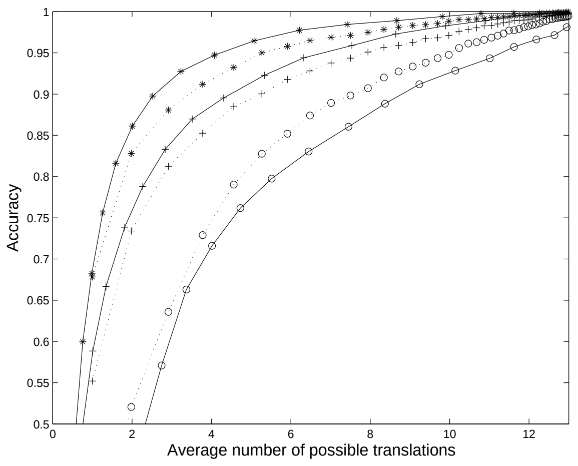


Figure 2: Accuracy of *best-n* strategy (dotted lines) and *cut-off* strategy (solid lines). o = generative alignment, + = discriminative alignment, * = word translation.

instance, to achieve 95% recall (that is, for 95% of the ambiguous words, we retain the correct translation), we only need candidate sets of average size 4.2 for the *cut-off* strategy using the word-translation model, whereas for the same strategy on the discriminative alignment model we require an average set size of 6.7 words.

As the size of the solution space grows exponentially with the size of the candidate sets, the word-translation model could potentially greatly reduce the search space while maintaining good accuracy.

It would be interesting to use similar techniques to learn null fertility (i.e., when a word a has no translation in the target sentence $\mathbf{t}$).

8 Related Work

Berger et al. (1996) apply maximum entropy methods (equivalent to logistic regression) to, among other tasks, the word-translation task. However, no quantitative results are presented. In this paper we demonstrate that the method can improve performance on a large data set and show how it might be used to improve machine translation.

Diab and Resnik (2002) suggest using large bilingual corpora to improve performance on word sense disambiguation. The main idea is that knowing a French word may help determine the meaning of the corresponding English word. They apply this intuition to the Senseval word disambiguation task by running off-the-shelf translators to produce translations which they then use for disambiguation.

Ng et al. (2003) address word sense disambiguation by manually annotating WordNet senses with their translation in the target language (Chinese), and then automatically extracting labeled examples for word sense disambiguation by applying the IBM

Models to a bilingual corpus. They achieve comparable results to training on hand-labeled examples.

Koehn and Knight (2003) focus on the task of noun-phrase translation. They improve performance on the noun-phrase translation task, and show that they can use this to improve full translations. A key difference is that, in predicting noun-phrase translations, they do not consider the context of nouns. They present results which indicate that humans can accurately translate noun phrases without looking at the surrounding context. However, as we have demonstrated in this paper, context can be very useful for a (sub-human-level) machine translator.

A similar argument applies to phrase-based translation methods (e.g., Koehn et al. (2003)). While phrase-based systems do take into account context within phrases, they are not able to use context across phrase boundaries. This is especially important when ambiguous words do not occur as part of a phrase — verbs in particular often appear alone.

9 Conclusions

In this paper, we focus on the word-translation problem. By viewing word-sense disambiguation in the context of a larger task, we were able to obtain large amounts of training data and directly evaluate the usefulness of our system for a real-world task. Our results improve over a baseline which is difficult to outperform in the word sense disambiguation task.

The word translation model could be improved in a variety of ways, drawing upon the large body of work on word-sense disambiguation. In particular, there are many types of context features which could be used to improve word translation performance, but which are not available to standard machine-translation systems. Also, the model could be extended to handle phrases.

To evaluate word translation in the context of a machine translation task, we introduce the novel blank-filling task, which decouples the impact of word translation from a variety of other factors, such as syntactic correctness. For this task, increased word-translation accuracy leads to improved machine translation. We also show that the word translation model is effective at choosing sets of candidate translations, suggesting that a word translation component would be immediately useful to current machine translations systems.

There are several ways in which the results of word translation could be integrated into a full translation system. Most naturally, the word translation model can be used directly to modify the score of different translations. Alternatively, a decoder can produce several candidate translations, which can be reranked using the word translation model. Unfortunately, we were unable to try these approaches, due to the lack of an appropriate publicly available decoder. Carpuat and Wu (2005) recently observed that simpler integration approaches, such as forcing the machine translation system to use the word translation model's first choice, do not improve translation results. Together, these results suggest that one should incorporate the results of word translation in a "soft" way, allowing the word translation, alignment, and language models to work together to produce coherent translations. Given an appropriate decoder, trying such a unified approach is straightforward, and would provide insight about the value of word translation.

References

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2).

M. Carpuat and D. Wu. 2005. Word sense disambiguation vs. statistical machine translation. *Proc. ACL*.

M. Diab and P. Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. *Proc. ACL*.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada. 2001. Fast decoding and optimal decoding for machine translation. *Proc. ACL*.

P. Koehn and K. Knight. 2003. Feature-rich statistical translation of noun phrases. *Proc. ACL*.

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. *HLT/NAACL*.

T. Minka. 2000. Algorithms for maximum-likelihood logistic regression. http://lib.stat.cmu.edu/ minka/papers/logreg.html.

A. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Proc. NIPS*.

H. T. Ng, B. Wang, and Y. S. Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. *Proc. ACL*.

F. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. *Proc. ACL*.

J. Shewchuk. 1994. An introduction to the conjugate gradient method without the agonizing pain. http://www-2.cs.cmu.edu/ jrs/jrspapers.html.

The Hiero Machine Translation System: Extensions, Evaluation, and Analysis

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, Michael Subotin
Institute for Advanced Computer Studies (UMIACS)
University of Maryland, College Park, MD 20742, USA
{dchiang,alopez,nmadnani,christof,resnik,msubotin}@umiacs.umd.edu

Abstract

Hierarchical organization is a well known property of language, and yet the notion of hierarchical structure has been largely absent from the best performing machine translation systems in recent community-wide evaluations. In this paper, we discuss a new hierarchical phrase-based statistical machine translation system (Chiang, 2005), presenting recent extensions to the original proposal, new evaluation results in a community-wide evaluation, and a novel technique for fine-grained comparative analysis of MT systems.

1 Introduction

Hierarchical organization is a well known property of language, and yet the notion of hierarchical structure has, for the last several years, been absent from the best performing machine translation systems in community-wide evaluations. Statistical phrase-based models (e.g. (Och and Ney, 2004; Koehn et al., 2003; Marcu and Wong, 2002)) characterize a source sentence f as a flat partition of non-overlapping subsequences, or "phrases", $\bar{f}_1 \cdots \bar{f}_J$, and the process of translation involves selecting target phrases $\bar{e}_i$ corresponding to the $\bar{f}_j$ and modifying their sequential order. The need for some way to model aspects of syntactic behavior, such as the tendency of constituents to move together as a unit, is widely recognized—the role of syntactic units is well attested in recent systematic studies of translation (Fox, 2002; Hwa et al., 2002; Koehn and Knight, 2003), and their absence in phrase-based models is quite evident when looking at MT system output. Nonetheless, attempts to incorporate richer linguistic features have generally met with little success (Och et al., 2004a).

Chiang (2005) introduces Hiero, a hierarchical phrase-based model for statistical machine translation. Hiero extends the standard, non-hierarchical notion of "phrases" to include nonterminal symbols, which permits it to capture both word-level and phrase-level reorderings within the same framework. The model has the formal structure of a synchronous CFG, but it does not make any commitment to a linguistically relevant analysis, and it does not require syntactically annotated training data. Chiang (2005) reported significant performance improvements in Chinese-English translation as compared with Pharaoh, a state-of-the-art phrase-based system (Koehn, 2004).

In Section 2, we review the essential elements of Hiero. In Section 3 we describe extensions to this system, including new features involving named entities and numbers and support for a fourfold scale-up in training set size. Section 4 presents new evaluation results for Chinese-English as well as Arabic-English translation, obtained in the context of the 2005 NIST MT Eval exercise. In Section 5, we introduce a novel technique for fine-grained comparative analysis of MT systems, which we employ in analyzing differences between Hiero's and Pharaoh's translations.

2 Hiero

Hiero is a stochastic synchronous CFG, whose productions are extracted automatically from unannotated parallel texts, and whose rule probabilities form a log-linear model learned by minimum-error-rate training; together with a modified CKY beam-search decoder (similar to that of Wu (1996)). We describe these components in brief below.

$$
\begin{aligned}
&\mathrm{S} \rightarrow \langle \mathrm{S}_{\boxed{1}}\mathrm{X}_{\boxed{2}}, \mathrm{S}_{\boxed{1}}\mathrm{X}_{\boxed{2}} \rangle \\
&\mathrm{S} \rightarrow \langle \mathrm{X}_{\boxed{1}}, \mathrm{X}_{\boxed{1}} \rangle \\
&\mathrm{X} \rightarrow \langle \text{yu } \mathrm{X}_{\boxed{1}} \text{ you } \mathrm{X}_{\boxed{2}}, \text{have } \mathrm{X}_{\boxed{2}} \text{ with } \mathrm{X}_{\boxed{1}} \rangle \\
&\mathrm{X} \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ de } \mathrm{X}_{\boxed{2}}, \text{the } \mathrm{X}_{\boxed{2}} \text{ that } \mathrm{X}_{\boxed{1}} \rangle \\
&\mathrm{X} \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ zhiyi}, \text{one of } \mathrm{X}_{\boxed{1}} \rangle \\
&\mathrm{X} \rightarrow \langle \text{Aozhou}, \text{Australia} \rangle \\
&\mathrm{X} \rightarrow \langle \text{shi}, \text{is} \rangle \\
&\mathrm{X} \rightarrow \langle \text{shaoshu guojia}, \text{few countries} \rangle \\
&\mathrm{X} \rightarrow \langle \text{bangjiao}, \text{diplomatic relations} \rangle \\
&\mathrm{X} \rightarrow \langle \text{Bei Han}, \text{North Korea} \rangle
\end{aligned}
$$

Figure 1: Example synchronous CFG

2.1 Grammar

A *synchronous CFG* or *syntax-directed transduction grammar* (Lewis and Stearns, 1968) consists of pairs of CFG rules with aligned nonterminal symbols. We denote this alignment by coindexation with boxed numbers (Figure 1). A derivation starts with a pair of aligned start symbols, and proceeds by rewriting pairs of aligned nonterminal symbols using the paired rules (Figure 2).

Training begins with phrase pairs, obtained as by Och, Koehn, and others: GIZA++ (Och and Ney, 2000) is used to obtain one-to-many word alignments in both directions, which are combined into a single set of refined alignments using the "final-and" method of Koehn et al. (2003); then those pairs of substrings that are exclusively aligned to each other are extracted as phrase pairs.

Then, synchronous CFG rules are constructed out of the initial phrase pairs by subtraction: every phrase pair $\langle \bar{f}, \bar{e} \rangle$ becomes a rule $\mathrm{X} \rightarrow \langle \bar{f}, \bar{e} \rangle$, and a phrase pair $\langle \bar{f}, \bar{e} \rangle$ can be subtracted from a rule $\mathrm{X} \rightarrow \langle \gamma_1 \bar{f} \gamma_2, \alpha_1 \bar{e} \alpha_2 \rangle$ to form a new rule $\mathrm{X} \rightarrow \langle \gamma_1 \mathrm{X}_{\boxed{i}} \gamma_2, \alpha_1 \mathrm{X}_{\boxed{i}} \alpha_2 \rangle$, where i is an index not already used. Various filters are also applied to reduce the number of extracted rules. Since one of these filters restricts the number of nonterminal symbols to two, our extracted grammar is equivalent to an inversion transduction grammar (Wu, 1997).

2.2 Model

The model is a log-linear model (Och and Ney, 2002) over synchronous CFG derivations. The weight of a derivation is $P_{LM}(e)^{\lambda_{LM}}$, the weighted language model probability, multiplied by the product of the weights of the rules used in the derivation. The weight of each rule is, in turn:

$$(1) \qquad w(X \rightarrow \langle \gamma, \alpha \rangle) = \prod_i \phi_i(X \rightarrow \langle \gamma, \alpha \rangle)^{\lambda_i}$$

where the ϕ_i are features defined on rules. The basic model uses the following features, analogous to Pharaoh's default feature set:

- $P(\gamma \mid \alpha)$ and $P(\alpha \mid \gamma)$
- the lexical weights $P_w(\gamma \mid \alpha)$ and $P_w(\alpha \mid \gamma)$ (Koehn et al., 2003);[1]
- a phrase penalty $\exp(1)$;
- a word penalty $\exp(l)$, where l is the number of terminals in α.

The exceptions to the above are the two "glue" rules, which are the rules with left-hand side S in Figure 1. The second has weight one, and the first has weight $w(\mathrm{S} \rightarrow \langle \mathrm{S}_{\boxed{1}}\mathrm{X}_{\boxed{2}}, \mathrm{S}_{\boxed{1}}\mathrm{X}_{\boxed{2}} \rangle) = \exp(-\lambda_g)$, the idea being that parameter λ_g controls the model's preference for hierarchical phrases over serial combination of phrases.

Phrase translation probabilities are estimated by relative-frequency estimation. Since the extraction process does not generate a unique derivation for each training sentence pair, a distribution over possible derivations is hypothesized, which gives uniform weight to all initial phrases extracted from a sentence pair and uniform weight to all rules formed out of an initial phrase. This distribution is then used to estimate the phrase translation probabilities.

The lexical-weighting features are estimated using a method similar to that of Koehn et al. (2003). The language model is a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998), trained using the SRI-LM toolkit (Stolcke, 2002).

[1]This feature uses word alignment information, which is discarded in the final grammar. If a rule occurs in training with more than one possible word alignment, Koehn et al. take the maximum lexical weight; Hiero uses a weighted average.

$$\begin{aligned}
\langle \mathrm{S}_{\boxed{1}}, \mathrm{S}_{\boxed{1}} \rangle &\Rightarrow \langle \mathrm{S}_{\boxed{2}} \mathrm{X}_{\boxed{3}}, \mathrm{S}_{\boxed{2}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \mathrm{S}_{\boxed{4}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \mathrm{S}_{\boxed{4}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \mathrm{X}_{\boxed{6}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \mathrm{X}_{\boxed{6}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \text{Aozhou } \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \text{Australia } \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi } \mathrm{X}_{\boxed{3}}, \text{Australia is } \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi } \mathrm{X}_{\boxed{7}} \text{ zhiyi, Australia is one of } \mathrm{X}_{\boxed{7}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi } \mathrm{X}_{\boxed{8}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi, Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that } \mathrm{X}_{\boxed{8}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi yu } \mathrm{X}_{\boxed{1}} \text{ you } \mathrm{X}_{\boxed{2}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi, Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that have } \mathrm{X}_{\boxed{2}} \text{ with } \mathrm{X}_{\boxed{1}} \rangle
\end{aligned}$$

Figure 2: Example partial derivation of a synchronous CFG.

The feature weights are learned by maximizing the BLEU score (Papineni et al., 2002) on held-out data, using minimum-error-rate training (Och, 2003) as implemented by Koehn. The implementation was slightly modified to ensure that the BLEU scoring matches NIST's definition and that hypotheses in the *n*-best lists are merged when they have the same translation and the same feature vector.

3 Extensions

In this section we describe our extensions to the base Hiero system that improve its performance significantly. First, we describe the addition of two new features to the Chinese model, in a manner similar to that of Och et al. (2004b); then we describe how we scaled the system up to a much larger training set.

3.1 New features

The LDC Chinese-English named entity lists (900k entries) are a potentially valuable resource, but previous experiments have suggested that simply adding them to the training data does not help (Vogel et al., 2003). Instead, we placed them in a supplementary phrase-translation table, giving greater weight to phrases that occurred less frequently in the primary training data. For each entry $\langle f, \{e_1, \ldots, e_n\}\rangle$, we counted the number of times $c(f)$ that f appeared in the primary training data, and assigned the entry the weight $\frac{1}{c(f)+1}$, which was then distributed evenly among the supplementary phrase pairs $\{\langle f, e_i\rangle\}$. We then created a new model feature for named entities. When one of these supplementary phrase pairs was used in translation, its feature value for the named-entity feature was the weight defined above, and its value in the other phrase-translation and lexical-weighting features was zero. Since these scores belonged to a separate feature from the primary translation probabilities, they could be reweighted independently during minimum-error-rate training.

Similarly, to process Chinese numbers and dates, we wrote a rule-based Chinese number/date translator, and created a new model feature for it. Again, the weight given to this module was optimized during minimum-error-rate training. In some cases we wrote the rules to provide multiple uniformly-weighted English translations for a Chinese phrase (for example, 八日 (*bari*) could become "the 8th" or "on the 8th"), allowing the language model to decide between the options.

3.2 Scaling up training

Chiang (2005) reports on experiments in Chinese-English translation using a model trained on 7.2M+9.2M words of parallel data.[2] For the NIST MT Eval 2005 large training condition, considerably more data than this is allowable. We chose to use only newswire data, plus data from Sinorama, a Taiwanese news magazine.[3] This amounts to almost 30M+30M words. Scaling to this set required reducing the initial limit on phrase lengths, previously fixed at 10, to avoid explosive growth of

[2]Here and below, the notation "X + Y words" denotes X words of foreign text and Y words of English text.

[3]From Sinorama, only data from 1991 and later were used, as articles prior to that were translated quite loosely.

the extracted grammar. However, since longer initial phrases can be beneficial for translation accuracy, we adopted a variable length limit: 10 for the FBIS corpus and other mainland newswire sources, and 7 for the HK News corpus and Sinorama. (During decoding, limits of up to 15 were sometimes used; in principle these limits should all be the same, but in practice it is preferable to tune them separately.)

For Arabic-English translation, we used the basic Hiero model, without special features for named entities or numbers/dates. We again used only the newswire portions of the allowable training data; we also excluded the Ummah data, as the translations were found to be quite loose. Since this amounted to only about 1.5M+1.5M words, we used a higher initial phrase limit of 15 during both training and decoding.

4 Evaluation

Figure 1 shows the performance of several systems on NIST MT Eval 2003 Chinese test data: Pharaoh (2004 version), trained only on the FBIS data; Hiero, with various combinations of the new features and the larger training data.[4] This table also shows Hiero's performance on the NIST 2005 MT evaluation task.[5] The metric here is case-sensitive BLEU.[6]

Figure 2 shows the performance of two systems on Arabic in the NIST 2005 MT Evaluation task: DC, a phrase-based decoder for a model trained by Pharaoh, and Hiero.

5 Analysis

Over the last few years, several automatic metrics for machine translation evaluation have been introduced, largely to reduce the human cost of iterative system evaluation during the development cycle (Lin and Och, 2004; Melamed et al., 2003; Papineni et al., 2002). All are predicated on the concept of *n*-gram matching between the sentence hypothesized by the translation system and one or more *reference translations*—that is, human translations for the test sentence. Although the motivations and formulae underlying these metrics are all different, ultimately they all produce a single number representing the "goodness" of the MT system output over a set of reference documents. This facility is valuable in determining whether a given system modification has a positive impact on overall translation performance. However, the metrics are all holistic. They provide no insight into the specific competencies or weaknesses of one system relative to another.

System	Features	Train	MT03	MT05
Pharaoh	standard	FBIS	0.268	
Hiero	standard	FBIS	0.288	
Hiero	standard	full	0.329	
Hiero	+nums, names	full	0.339	0.300

Table 1: Chinese results. (BLEU-4; MT03 case-insensitive, MT05 case-sensitive)

System	Train	MT05
DC	full	0.399
Hiero	full	0.450

Table 2: Arabic results. (BLEU-4; MT03 case-insensitive, MT05 scores case-sensitive.

Ideally, we would like to use automatic methods to provide immediate diagnostic information about the translation output—*what* the system does well, and what it does poorly. At the most general level, we want to know how our system performs on the two most basic problems in translation—word translation and reordering. Unigram precision and recall statistics tell us something about the performance of an MT system's internal translation dictionaries, but nothing about reordering. It is thought that higher order *n*-grams correlate with the reordering accuracy of MT systems, but this is again a holistic metric.

What we would really like to know is how well the system is able to capture systematic reordering patterns in the input, which ones it is successful with, and which ones it has difficulty with. Word *n*-grams are little help here: they are too many, too sparse, and it is difficult to discern general patterns from them.

[4]The third line, corresponding to the model without new features trained on the larger data, may be slightly depressed because the feature weights from the fourth line were used instead of doing minimum-error-rate training specially for this model.

[5]Full results are available at `http://www.nist.gov/speech/tests/summaries/2005/mt05.htm`. For this test, a phrase length limit of 15 was used during decoding.

[6]For this task, the translation output was uppercased using the SRI-LM toolkit: essentially, it was decoded again using an HMM whose states and transitions are a trigram language model of cased English, and whose emission probabilities are reversed, i.e., probability of cased word given lowercased word.

5.1 A New Analysis Method

In developing a new analysis method, we are motivated in part by recent studies suggesting that word reorderings follow general patterns with respect to syntax, although there remains a high degree of flexibility (Fox, 2002; Hwa et al., 2002). This suggests that in a comparative analysis of two MT systems, it may be useful to look for syntactic patterns that one system captures well in the target language and the other does not, using a syntax based metric.

We propose to summarize reordering patterns using part-of-speech sequences. Unfortunately, recent work has shown that applying statistical parsers to ungrammatical MT output is unreliable at best, with the parser often assigning unreasonable probabilities and incongruent structure (Yamada and Knight, 2002; Och et al., 2004a). Anticipating that this would be equally problematic for part-of-speech tagging, we make the conservative choice to apply annotation only to the reference corpus. Word *n*-gram correspondences with a reference translation are used to infer the part-of-speech tags for words in the system output.

First, we tagged the reference corpus with parts of speech. We used MXPOST (Ratnaparkhi, 1996), and in order to discover more general patterns, we map the tag set down after tagging, e.g. NN, NNP, NNPS and NNS all map to NN. Second, we computed the frequency $\mathit{freq}(t_i \ldots t_j)$ of every possible tag sequence $t_i \ldots t_j$ in the reference corpus. Third, we computed the correspondence between each hypothesis sentence and *each* of its corresponding reference sentences using an approximation to maximum matching (Melamed et al., 2003). This algorithm provides a list of *runs* or contiguous sequences of words $e_i \ldots e_j$ in the reference that are also present in the hypothesis. (Note that runs are order-sensitive.) Fourth, for each recalled *n*-gram $e_i \ldots e_j$, we looked up the associated tag sequence $t_i \ldots t_j$ and incremented a counter $\mathit{recalled}(t_i \ldots t_j)$. Finally, we computed the recall of tag patterns, $R(t_i \ldots t_j) = \mathit{recalled}(t_i \ldots t_j)/\mathit{freq}(t_i \ldots t_j)$, for all patterns in the corpus.

By examining examples of these tag sequences in the reference corpus and their hypothesized translations, we expect to gain some insight into the comparative strengths and weaknesses of the MT systems' reordering models. (An interactive platform for this analysis is demonstrated by Lopez and Resnik (2005).)

5.2 Chinese

We performed tag sequence analysis on the Hiero and Pharaoh systems trained on the FBIS data only. Table 3 shows those *n*-grams for which Hiero and Pharaoh's recall differed significantly ($p < 0.01$). The numbers shown are the ratio of Hiero's recall to Pharaoh's. Note that the *n*-grams on which Hiero had better recall are dominated by fragments of prepositional phrases (in the Penn Treebank tagset, prepositions are tagged IN or TO).

Our hypothesis is that Hiero produces English PPs better because many of them are translated from Chinese phrases which have an NP modifying an NP to its right, often connected with the particle 的 (*de*). These are often translated into English as PPs, which modify the NP to the left. A correct translation, then, would have to reorder the two NPs. Notice in the table that Hiero recalls proportionally more *n*-grams as *n* increases, corroborating the intuition that Hiero should be better at longer-distance reorderings.

Investigating this hypothesis qualitatively, we inspected the first five occurrences of the *n*-grams of the first type on the list (JJ NN IN DT NN). Of these, we omit one example because both systems recalled the *n*-gram correctly, and one because they differed only in lexical choice (Hiero matched the 5-gram with one reference sentence, Pharaoh with zero). The other three examples are shown below (H = Hiero, P = Pharaoh):

(2)

联合国	安全	理事会	的	五个	常任
UN	security	council	of	five	permanent
理事	国都				
member	countries-all				

R1. five permanent members of the UN Security Council

H. the five permanent members of the un security council

P. the united nations security council permanent members of the five countries

10.00	JJ NN IN DT NN	3.14	DT NN IN DT NN	1.46	DT NN PU	1.09	CD
7.00	IN NN TO	3.00	IN DT NN PU	1.44	IN DT JJ	1.07	VB
5.50	IN DT NN NN PU NN	2.50	NN TO NN	1.42	NN IN DT	1.06	NN NN
5.50	IN DT NN NN PU NN NN	2.03	DT JJ NN IN	1.41	IN DT NN	1.06	IN
4.50	NN JJ NN PU	1.95	IN NN PU	1.37	PU CC	1.05	NN
4.50	NN IN DT JJ	1.77	IN NN CD	1.34	IN CD	0.61	RB CD
4.00	VB CD IN DT	1.74	DT NN IN NN	1.32	JJ NN PU	0.21	TO VB PR
3.67	IN DT NN NN PU	1.70	JJ NN IN	1.30	IN NN	0.18	PU RB CD
3.50	NN IN DT NN NN	1.55	VB IN DT	1.29	NN IN	0.12	NN CD TO NN
3.30	NN IN DT NN	1.46	NN IN NN	1.18	NN PU	0.12	CD TO NN

Table 3: Chinese-English POS *n*-grams on which Hiero and Pharaoh had significantly different recall, arranged by recall ratio. Ratio > 1 indicates tag sequences that Hiero matched more frequently.

(3) 伊拉克 危机 的 最 新 发展
Iraq crisis of most new development

R1. the latest development on the Iraqi crisis
H. the latest development on the Iraqi crisis
P. on the iraqi crisis, the latest development

(4) 今年 上 半年
this-year upper half-year

R1. the first half of this year
H. the first half of this year
P. the first half of

All three of these examples involve an NP modifying an NP to its right; two with the particle 的 (*de*) and one without. In all three cases Hiero reorders the NPs correctly; Pharaoh preserves the Chinese word order in two cases, but in the third, for reasons not understood, drops the modifying NP.

The *n*-grams on which Hiero did worse than Pharaoh mostly involved numbers; here a pattern is not as easily discernible, but there are several cases where Hiero makes errors in translating numbers (neither system in this comparison used the dedicated number translator). For the *n*-gram TO VB PR, it seems Hiero has a tendency to delete possessive pronouns (PR, abbreviated from PRP$).

5.3 Arabic

Initial inspection of the *n*-grams on which Hiero showed significantly higher recall in the Arabic-English task suggested that here, too, better translation of nominal phrases may be at play. We investigated this conjecture further by examining several *n*-gram sets with the highest recall ratios. Some of them on closer inspection turned out to conflate different structural patterns, and provided little interpretable information. However, the 8 sentences in the *n*-gram list IN DT JJ JJ showed a degree of structural consistency. The list contained 6 instances where Hiero performed better in translating a complex NP or PP, one instance in which DC performed better in translating a complex PP, and one case in which they both performed equally poorly. Below we show two examples of phrases on which Hiero performed better, and the one example on which its hierarchical approach produced undesirable results (H = Hiero, D = DC).

(5) Al wjwd Al EskrY Al AmYrkY fY Al mnTqp
the presence the military the American in the region

R1. the American military presence in the region
H. the american military presence in the region
D. the military presence in the region

(6) AltY tSnEhA Al $rkp Al kwrYp Al jnwbYp
which manufactures-them the company the Korean the Southern

R1. which are manufactured by the South Korean company
H. which are manufactured by the south korean company
D. which are manufactured by the company , the south korean

8.00	WR DT NN	2.38	DT JJ JJ NN	1.46	JJ NN NN		
8.00	PR NN IN DT	2.08	CC JJ NN	1.43	JJ JJ	1.02	NN
7.00	DT PU	2.01	PR VB	1.35	IN DT JJ	0.47	NN CD PU CD NN NN
6.00	DT NN NN PO	2.00	TO DT NN NN	1.24	VB IN	0.47	NN CD PU CD NN NN NN
5.00	IN DT JJ JJ	1.80	NN PU WD	1.21	NN VB	0.47	NN CD PU CD NN NN NN PU
4.67	DT NN IN VB	1.80	NN IN DT JJ NN	1.20	NN IN DT	0.45	NN CD PU CD NN
2.89	NN NN NN VB	1.77	NN IN DT JJ	1.17	PR	0.29	NN CD NN
2.73	PR VB IN	1.76	JJ JJ NN	1.10	JJ NN	0.27	NN CD NN CD
2.56	NN PU WD VB	1.74	VB CD	1.08	NN NN	0.09	NN CD NN PU
2.45	JJ CC JJ NN	1.68	NN NN VB	1.07	IN DT		

Table 4: Arabic-English POS *n*-grams on which Hiero and DC had significantly different recall, arranged by recall ratio. Ratio > 1 indicates tag sequences that Hiero matched more frequently.

(7) swq Al EqArAt fY Akbr mdYnp
market the real-estate in largest city
SnAEYp SYnYp $AnghAY
industrial Chinese Shanghai

R2. The real estate market in the largest Chinese industrial city , Shanghai

H. chinese real estate market in the largest industrial city shanghai

D. real estate market in the largest chinese industrial city shanghai

In the last example we see that Hiero mistakenly identified the adjective "Chinese" as modifying the highest head of the first NP in the apposition.

The style of Arabic newswire tends strongly towards the verb-initial word order in the main clause. Based on our inspection of the *n*-gram collection NN VB, we were also able to note that Hiero performed noticeably better in reordering the subject and main verb to produce idiomatic English translations. Although in this set the differences in the recall for the NN VB bigram were influenced by many different translation issues, reordering the subject and main verbs was the only structural pattern that recurred consistently throughout the set, appearing in 8 of the 29 relevant sentences.

(8) wqAl Al bYAn An
and-said the statement that

R1. The statement said

H. the statement said that

D. said a statement that

(9) AEln ms&wl fY Al Amm Al mtHdp An
announced official in the nations the united that

R1. A United Nations official announced that

H. the united nations official announced that

D. an official in the united nations that

Looking at the bottom of the list, we find more examples of how Hiero's reordering behavior sometimes backfires. These *n*-grams seem primarily to be parts of bylines, where Hiero has a tendency to reformat the date, whereas DC keeps the original format, matching more often.

(10) mAnYlA 26 YnAYr
Manila 26 January

R3. Manila 26 January

H. manila , january 26

P. manila 26 january

6 Conclusions

The work reported in this paper extends the original treatment of Hiero (Chiang, 2005) by evaluating an improved version in a community-wide exercise for Chinese-English and Arabic-English translation, and by introducing a novel analysis technique for comparing MT systems' output. The evaluation results provide strong evidence that the approach gains performance from its hierarchical extensions to phrase-based translation. The analysis of part-of-speech tag sequences provides a way to perform finer-grained comparison of system output, pinpointing phenomena for which the systems differ significantly.

Acknowledgements

We would like to thank Philipp Koehn for the use of the Pharaoh system. This research was supported in part by ONR MURI Contract FCPO.810548265 and Department of Defense contract RD-02-5700.

References

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP 2002*, pages 304–311.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 392–399.

Philipp Koehn and Kevin Knight. 2003. Feature-rich statistical translation of noun phrases. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 311–318.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA 2004*, pages 115–124.

P. M. Lewis II and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15:465–488.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 606–613.

Adam Lopez and Philip Resnik. 2005. Pattern visualization for machine translation output. In *Proceedings of HLT/EMNLP 2005*. Demonstration session.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP 2002*, pages 133–139.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. 2003. Precision and recall of machine translation. In *Proceedings of HLT-NAACL 2003*, pages 61–63.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 440–447.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 295–302.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004a. A smorgasbord of features for statistical machine translation. In *Proceedings of HLT-NAACL 2004*.

Franz Josef Och, Ignacio Thayer, Daniel Marcu, Kevin Knight, Dragos Stefan Munteanu, Quamrul Tipu, Michel Galley, and Mark Hopkins. 2004b. Arabic and Chinese MT at USC/ISI. Presentation given at NIST Machine Translation Evaluation Workshop.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318.

Adwait Ratnaparkhi. 1996. A maximum-entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. 2003. The CMU statistical machine translation system. In *Proceedings of MT-Summit IX*, pages 402–409.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 152–158.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 303–310.

Comparing and Combining Finite-State and Context-Free Parsers

Kristy Hollingshead and **Seeger Fisher** and **Brian Roark**
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
Beaverton, Oregon, 97006
{hollingk,fishers,roark}@cslu.ogi.edu

Abstract

In this paper, we look at comparing high-accuracy context-free parsers with high-accuracy finite-state (shallow) parsers on several shallow parsing tasks. We show that previously reported comparisons greatly under-estimated the performance of context-free parsers for these tasks. We also demonstrate that context-free parsers can train effectively on relatively little training data, and are more robust to domain shift for shallow parsing tasks than has been previously reported. Finally, we establish that combining the output of context-free and finite-state parsers gives much higher results than the previous-best published results, on several common tasks. While the efficiency benefit of finite-state models is inarguable, the results presented here show that the corresponding cost in accuracy is higher than previously thought.

1 Introduction

Finite-state parsing (also called chunking or shallow parsing) has typically been motivated as a fast first-pass for – or approximation to – more expensive context-free parsing (Abney, 1991; Ramshaw and Marcus, 1995; Abney, 1996). For many very-large-scale natural language processing tasks (e.g. open-domain question answering from the web), context-free parsing may be too expensive, whereas finite-state parsing is many orders of magnitude faster and can also provide very useful syntactic annotations for large amounts of text. For this reason, finite-state parsing (hereafter referred to as shallow parsing) has received increasing attention in recent years.

In addition to the clear efficiency benefit of shallow parsing, Li and Roth (2001) have further claimed both an accuracy and a robustness benefit versus context-free parsing. The output of a context-free parser, such as that of Collins (1997) or Charniak (2000), can be transformed into a sequence of shallow constituents for comparison with the output of a shallow parser. Li and Roth demonstrated that their shallow parser, trained to label shallow constituents along the lines of the well-known CoNLL-2000 task (Sang and Buchholz, 2000), outperformed the Collins parser in correctly identifying these constituents in the Penn Wall Street Journal (WSJ) Treebank (Marcus et al., 1993). They argued that their superior performance was due to optimizing directly for the local sequence labeling objective, rather than for obtaining a hierarchical analysis over the entire string. They further showed that their shallow parser trained on the Penn WSJ Treebank did a far better job of annotating out-of-domain sentences (e.g. conversational speech) than the Collins parser.

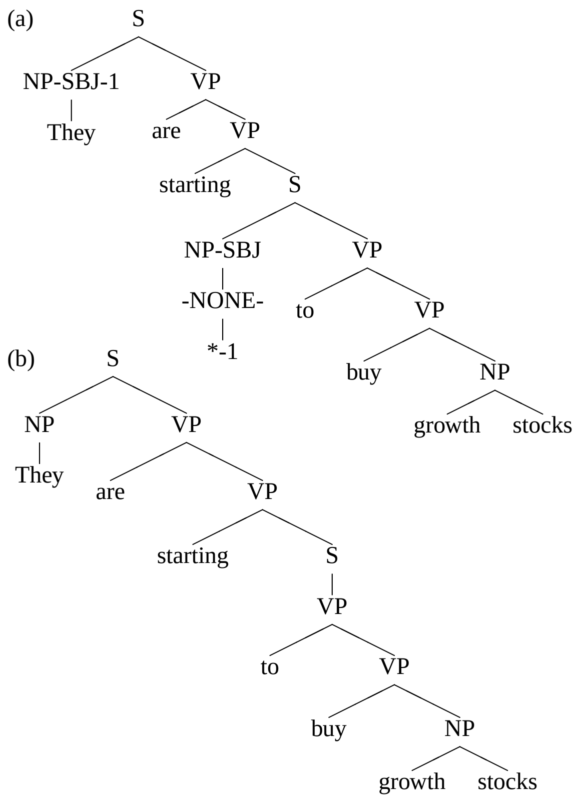
This paper re-examines the comparison of shallow parsers with context-free parsers, beginning with a critical examination of how their outputs are compared. We demonstrate that changes to the conversion routine, which take into account differences between the original treebank trees and the trees output by context-free parsers, eliminate the previously-reported accuracy differences. Second, we show that a convention that is widely accepted for evaluation of context-free parses – ignoring punctuation when setting the span of a constituent – results in improved shallow parsing performance by certain context-free parsers across a variety of shallow parsing tasks. We also demonstrate that context-free parsers perform competitively when applied to out-of-domain data. Finally, we show that large improvements can be obtained in several shallow parsing tasks by using simple strategies to incorporate context-free parser output into shallow parsing models. Our results demonstrate that a rich context-free parsing model is, time permitting, worth applying, even if only shallow parsing output is needed. In addition, our best results, which greatly improve on the previous-best published results on several tasks, shed light on how much accuracy is sacrificed in shallow parsing to get finite-state efficiency.

2 Evaluating Heterogeneous Parser Output

Two commonly reported shallow parsing tasks are Noun-Phrase (NP) Chunking (Ramshaw and Marcus, 1995) and the CoNLL-2000 Chunking task (Sang and Buchholz, 2000), which extends the NP-Chunking task to recognition of 11 phrase types[1] annotated in the Penn Treebank. Reference shallow parses for this latter task were derived from treebank trees via a conversion script known as `chunklink`[2]. We follow Li and Roth (2001) in using `chunklink` to also convert trees output by a context-free parser into a flat representation of shallow constituents. Figure 1(a) shows a Penn Treebank tree and Figure 1(c) its corresponding shallow parse constituents, according to the CoNLL-2000 guidelines. Note that consecutive verb phrase (VP) nodes result in a single VP shallow constituent.

Just as the original treebank trees are converted for training shallow parsers, they are also typically modified for training context-free parsers. This modification includes removal of empty nodes (nodes tagged with "-NONE-" in the treebank), and removal of function tags on non-terminals; e.g., NP-SBJ (subject NP) and NP-TMP (temporal NP) are both mapped to NP. The output of the context-free parser is, of course, in the same format as the training input, so empty nodes and function tags are not present. This type of modified tree is what is shown in Figure 1(b); note that the original treebank tree, shown in Figure 1(a), had an empty subject NP in the embedded clause which has been removed for the modified tree.

To compare the output of their shallow parser with the output of the well-known Collins (1997) parser, Li and Roth applied the `chunklink` conversion script to extract the shallow constituents from the output of the Collins parser on WSJ section 00. Un-



(c) [NP They] [VP are starting to buy] [NP growth stocks]

Figure 1: (a) Penn WSJ treebank tree, (b) modified treebank tree, and (c) CoNLL-2000 style shallow bracketing, all of the same string.

fortunately, the script was built to be applied to the original treebank trees, complete with empty nodes, which are not present in the output of the Collins parser, or any well-known context-free parser. The `chunklink` script searches for empty nodes in the parse tree to perform some of its operations. In particular, any S node that contains an empty subject NP and a VP is reduced to just a VP node, and then combined with any immediately-preceding VP nodes to create a single VP constituent. If the S node does not contain an empty subject NP, as in Figure 1(b), the `chunklink` script creates two VP constituents: [VP are starting] [VP to buy], which in this case results in a bracketing error. However, it is a simple matter to insert an empty subject NP into unary S$\rightarrow$VP productions so that these nodes are processed correctly by the script.

Various conventions have become standard in evaluating parser output over the past decade. Perhaps the most widely accepted convention is that of ignoring punctuation for the purposes of assigning constituent span, under the perspective that, fun-

[1]These include: ADJP, ADVP, CONJP, INTJ, LST, NP, PP, PRT, SBAR, UCP and VP. Anything not in one of these base phrases is designated as "outside".

[2]Downloaded from http://ilk.kub.nl/∼sabine/chunklink/.

System	Phrase Type	Evaluation Scenario (a)	(b)	(c)
"Modified" Truth	All	98.37	99.72	99.72
	VP	92.14	98.70	98.70
Li and Roth (2001)	All	94.64	-	-
	VP	95.28	-	-
Collins (1997)	All	92.16	93.42	94.28
	VP	88.15	94.31	94.42
Charniak (2000)	All	93.88	95.15	95.32
	VP	88.92	95.11	95.19

Table 1: F-measure shallow bracketing accuracy under three different evaluation scenarios: (a) baseline, used in Li and Roth (2001), with original `chunklink` script converting treebank trees and context-free parser output; (b) same as (a), except that empty subject NPs are inserted into every unary S→VP production; and (c) same as (b), except that punctuation is ignored for setting constituent span. Results for Li and Roth are reported from their paper. The Collins parser is provided with part-of-speech tags output by the Brill tagger (Brill, 1995).

damentally, constituents are groupings of words. Interestingly, this convention was not followed in the CoNLL-2000 task (Sang and Buchholz, 2000), which as we will see has a variable effect on context-free parsers, presumably depending on the degree to which punctuation is moved in training.

2.1 Evaluation Analysis

To determine the effects of the conversion routine and different evaluation conventions, we compare the performance of several different models on one of the tasks presented in Li and Roth (2001). For this task, which we label the *Li & Roth task*, sections 2-21 of the Penn WSJ Treebank are used as training data, section 24 is held out, and section 00 is for evaluation.

For all trials in this paper, we report F-measure labeled bracketing accuracy, which is the harmonic mean of the labeled precision (P) and labeled recall (R), as they are defined in the widely used PARSEVAL metrics; i.e. the F-measure accuracy is $\frac{2PR}{P+R}$.

Table 1 shows baseline results for the Li and Roth[3] shallow parser, two well-known, high-accuracy context-free parsers, and the reference (true) parses after being modified as described above (by removing empty nodes and function tags). Evaluation scenario (a) in Table 1 corresponds to what was used in Li and Roth (2001) following CoNLL-2000 guidelines, with the original `chunklink` script used to transform the context-free parser output into shallow constituents. We can see from the performance of the modified truth in this scenario that there are serious problems with this conversion, due to the way in which it handles unary S→VP productions. If we deterministically insert empty subject NP nodes for all such unary productions prior to the use of the `chunklink` script, which we do in evaluation scenario (b) of Table 1, this repairs the bulk of the errors. Some small number of errors remain, due largely to the fact that if the S node has been annotated with a function tag (e.g. S-PRP, S-PRD, S-CLR), then `chunklink` will not perform its reduction operation on that node. However, for our purposes, this insertion repair sufficiently corrects the error to perform meaningful comparisons. Finally, evaluation scenario (c) follows the context-free parsing evaluation convention of ignoring punctuation when assigning constituent span. This affects some parsers more than others, depending on how the parser treats punctuation internally; for example, Bikel (2004) documents that the Collins parser raises punctuation nodes within the parse tree. Since ignoring punctuation cannot hurt performance, only improve it, even the smallest of these differences are statistically significant.

Note that after inserting empty nodes and ignoring punctuation, the accuracy advantage of Li and Roth over Collins is reduced to a dead heat. Of the two parsers we evaluated, the Charniak (2000) parser gave the best performance, which is consistent with its higher reported performance on the context-free parsing task versus other context-free parsers. Collins (2000) reported a reranking model that improved his parser output to roughly the same level of accuracy as Charniak (2000), and Charniak and Johnson (2005) report an improvement using reranking over Charniak (2000). For the purposes of this paper, we needed an available parser that was (a) trainable on different subsets of the data to be applied to various tasks; and (b) capable of producing n-best candidates, for potential combination with a shallow parser. Both the Bikel (2004) imple-

[3]We were unable to obtain the exact model used in Li and Roth (2001), and so we use their reported results here. Note that they used reference part-of-speech (POS) tags for their results on this task. All other results reported in this paper, unless otherwise noted, were obtained using Brill-tagger POS tags.

System		NP-Chunking	CoNLL-2000	Li & Roth task
SPRep averaged perceptron		94.21	93.54	95.12
Kudo and Matsumoto (2001)		94.22	93.91	-
Sha and Pereira (2003)	CRF	94.38	-	-
	Voted perceptron	94.09	-	-
Zhang et al. (2002)		-	94.17	-
Li and Roth (2001)		-	93.02	94.64

Table 2: Baseline results on three shallow parsing tasks: the NP-Chunking task (Ramshaw and Marcus, 1995); the CoNLL-2000 Chunking task (Sang and Buchholz, 2000); and the Li & Roth task (Li and Roth, 2001), which is the same as CoNLL-2000 but with more training data and a different test section. The results reported in this table include the best published results on each of these tasks.

mentation of the Collins parser and the n-best version of the Charniak (2000) parser, documented in Charniak and Johnson (2005), fit the requirements. Since we observed higher accuracy from the Charniak parser, from this point forward we report just Charniak parser results[4].

2.2 Shallow Parser

In addition to the trainable n-best context-free parser from Charniak (2000), we needed a trainable shallow parser to apply to the variety of tasks we were interested in investigating. To this end, we replicated the NP-chunker described in Sha and Pereira (2003) and trained it as either an NP-chunker or with the tagset extended to classify all 11 phrase types included in the CoNLL-2000 task (Sang and Buchholz, 2000). Our shallow parser uses exactly the feature set delineated by Sha and Pereira, and performs the decoding process using a Viterbi search with a second-order Markov assumption as they described. These features include unigram and bigram words up to two positions to either side of the current word; unigram, bigram, and trigram part-of-speech (POS) tags up to two positions to either side of the current word; and unigram, bigram, and trigram shallow constituent tags. We use the averaged perceptron algorithm, as presented in Collins (2002), to train the parser. See (Sha and Pereira, 2003) for more details on this approach.

To demonstrate the competitiveness of our baseline shallow parser, which we label the *SPRep averaged perceptron*, Table 2 shows results on three different shallow parsing tasks. The NP-Chunking task, originally introduced in Ramshaw and Marcus (1995) and also described in (Collins, 2002; Sha and Pereira, 2003), brackets just base NP constituents[5]. The CoNLL-2000 task, introduced as a shared task at the CoNLL workshop in 2000 (Sang and Buchholz, 2000), extends the NP-Chunking task to label 11 different base phrase constituents. For both of these tasks, the training set was sections 15-18 of the Penn WSJ Treebank and the test set was section 20. We follow Collins (2002) and Sha and Pereira (2003) in using section 21 as a heldout set. The third task, introduced by Li and Roth (2001), performs the same labeling as in the CoNLL-2000 task, but with more training data and different testing sets: training was WSJ sections 2-21 and test was section 00. We used section 24 as a heldout set; this section is often used as heldout for training context-free parsers.

Training and testing data for the CoNLL-2000 task is available online[6]. For the heldout sets for each of these tasks, as well as for all data sets needed for the Li & Roth task, reference shallow parses were generated using the `chunklink` script on the original treebank trees. All data was tagged with the Brill POS tagger (Brill, 1995) after the `chunklink` conversion. We verified that using this method on the original treebank trees in sections 15-18 and 20 generated data that is identical to the CoNLL-2000 data sets online. Replacing the POS tags in the input text with Brill POS tags before the

[4]The parser is available for research purposes at ftp://ftp.cs.brown.edu/pub/nlparser/ and can be run in n-best mode. The one-best performance of the parser is the same as what was presented in Charniak (2000).

[5]We follow Sha and Pereira (2003) in deriving the NP constituents from the CoNLL-2000 data sets, by replacing all non-NP shallow tags with the "outside" ("O") tag. They mention that the resulting shallow parse tags are somewhat different than those used by Ramshaw and Marcus (1995), but that they found no significant accuracy differences in training on either set.

[6]Downloaded from the CoNLL-2000 Shared Task website http://www.cnts.ua.ac.be/conll2000/chunking/.

`chunklink` conversion results in slightly different shallow parses.

From Table 2 we can see that our shallow parser is competitive on all three tasks[7]. Sha and Pereira (2003) noted that the difference between their perceptron and CRF results was not significant, and our performance falls between the two, thus replicating their result within noise. Our performance falls 0.6 percentage points below the best published result on the CoNLL-2000 task, and 0.5 percentage points above the performance by Li and Roth (2001) on their task. Overall, ours is a competitive approach for shallow parsing.

3 Experimental Results

3.1 Comparing Finite-State and Context-Free Parsers

The first two rows of Table 3 present a comparison between the SPRep shallow parser and the Charniak (2000) context-free parser detailed in Charniak and Johnson (2005). We can see that the performance of the two models is virtually indistinguishable for all three of these tasks, with or without ignoring of punctuation. As mentioned earlier, we used the version of this parser with improved n-best extraction, as documented in Charniak and Johnson (2005), although without the reranking of the candidates that they also report in that paper. For these trials, we used just the one-best output of that model, which is the same as in Charniak (2000).

Note that the standard training set for context-free parsing (sections 2-21) is only used for the Li & Roth task; for the other two tasks, both the SPRep and the Charniak parsers were trained on sections 15-18, with section 21 as heldout. This demonstrates that the context-free parser, even when trained on a small fraction of the total treebank, is able to learn a competitive model for this task.

3.2 Combining Finite-State and Context-Free Parsers

It is likely true that a context-free parser which has been optimized for global parse accuracy will, on occasion, lose some shallow parse accuracy to satisfy global structure constraints that do not constrain a shallow parser. However, it is also likely true that these longer distance constraints will on occasion enable the context-free parser to better identify the shallow constituent structure. In other words, despite having very similar performance, our shallow parser and the Charniak context-free parser are likely making complementary predictions about the shallow structure that can be exploited for further improvements. In this section, we explore two simple methods for combining the system outputs.

The first combination of the system outputs, which we call *unweighted intersection*, is the simplest kind of 'rovered' system, which restricts the set of shallow parse candidates to the intersection of the sets output by each system, but does not combine the scores. Since the Viterbi search of the SPRep model provides a score for all possible shallow parses, the intersection of the two sets is simply the set of shallow-parse sequences in the 50-best candidates output by the Charniak parser. We then use the SPRep perceptron-model scores to choose from among just these candidates. We converted the 50-best lists returned by the Charniak parser into k-best lists of shallow parses by using `chunklink` to convert each candidate context-free parse into a shallow parse. Many of the context-free parses map to the same shallow parse, so the size of this list is typically much less than 50, with an average of around 7. Each of the unique shallow-parse candidates is given a score by the SPRep perceptron, and the best-scoring candidate is selected. Effectively, we used the Charniak parser's k-best shallow parses to limit the search space for our shallow parser.

The second combination of the system outputs, which we call *weighted intersection*, extends the unweighted intersection by including the scores from the Charniak parser, which are log probabilities. The score for a shallow parse output by the Charniak parser is the log of the sum of the probabilities of all context-free parses mapping to that shallow parse. We normalize across all candidates for a given string, hence these are conditional log probabilities. We multiply these conditional log probabilities by a scaling factor α before adding them to the SPRep perceptron score for a particular candidate. Again, the best-scoring candidate using this composite score is selected from among the shallow

[7]Sha and Pereira (2003) reported the Kudo and Matsumoto (2001) performance on the NP-Chunking task to be 94.39 and to be the best reported result on this task. In the cited paper, however, the result is as reported in our table.

	NP-Chunking		CoNLL-2000		Li & Roth task	
	Punctuation		Punctuation		Punctuation	
System	Leave	Ignore	Leave	Ignore	Leave	Ignore
SPRep averaged perceptron	94.21	94.25	93.54	93.70	95.12	95.27
Charniak (2000)	94.17	94.20	93.77	93.92	95.15	95.32
Unweighted intersection	95.13	95.16	94.52	94.64	95.77	95.92
Weighted intersection	95.57	95.58	95.03	95.16	96.20	96.33

Table 3: F-measure shallow bracketing accuracy on three shallow parsing tasks, for the SPRep perceptron shallow parser, the Charniak (2000) context-free parser, and for systems combining the SPRep and Charniak system outputs.

parse candidates output by the Charniak parser. We used the heldout data to empirically estimate an optimal scaling factor for the Charniak scores, which is 15 for all trials reported here. This factor compensates for differences in the dynamic range of the scores of the two parsers.

Both of these intersections are done at test-time, i.e. the models are trained independently. To remain consistent with task-specific training and testing section conventions, the individual models were always trained on the appropriate sections for the given task, i.e. WSJ sections 15-18 for NP-Chunking and the CoNLL-2000 tasks, and sections 2-21 for the Li & Roth task.

Results from these methods of combination are shown in the bottom two rows of Table 3. Even the simple unweighted intersection gives quite large improvements over each of the independent systems for all three tasks. All of these improvements are significant at $p < 0.001$ using the Matched Pair Sentence Segment test (Gillick and Cox, 1989). The weighted intersection gives further improvements over the unweighted intersection for all tasks, and this improvement is also significant at $p < 0.001$, using the same test.

3.3 Robustness to Domain Shift

Our final shallow parsing task was also proposed in Li and Roth (2001). The purpose of this task was to examine the degradation in performance when parsers, trained on one relatively clean domain such as WSJ, are tested on another, mismatched domain such as Switchboard. The systems that are reported in this section are trained on sections 2-21 of the WSJ Treebank, with section 24 as heldout, and tested on section 4 of the Switchboard Treebank. Note that the systems used here are exactly the ones presented for the original Li & Roth task, in Sections 3.1 and 3.2; only the test set has changed, training and heldout sets remain exactly the same, as do the mixing parameters for the weighted intersection. In the trials reported in Li and Roth (2001), both of the evaluated systems were provided with reference POS tags from the Switchboard Treebank. In the current results, we show our SPRep averaged perceptron system provided both with reference POS tags for comparison with the Li and Roth results, and provided with Brill-tagger POS tags for comparison with other systems. Table 4 shows our results for this task. Whereas Li and Roth reported a more marked degradation in performance when using a context-free parser as compared to a shallow parser, we again show virtually indistinguishable performance between our SPRep shallow parser and the Charniak context-free parser. Again, using a weighted combined model gave us large improvements over each independent model, even in this mismatched domain.

	Punctuation	
System	Leave	Ignore
Li & Roth (reference tags)	88.47	-
SPRep avg perceptron		
Reference tags	91.37	91.86
Brill tags	87.94	88.42
Charniak (2000)	87.94	88.44
Unweighted intersection	88.66	89.16
Weighted intersection	89.22	89.69

Table 4: Shallow bracketing accuracy of several different systems, trained on sections 2-21 of WSJ Treebank and applied to section 4 of the Switchboard Treebank. Li and Roth (2001) results are as reported in their paper, with reference POS tags rather than Brill-tagger POS tags.

3.4 Reranked n-best List

Just prior to the publication of this paper, we were able to obtain the trained reranker from Charniak

System		WSJ Sect. 00		SWBD Sect. 4	
		Punctuation		Punctuation	
		Leave	Ignore	Leave	Ignore
SPRep		95.12	95.27	87.94	88.43
C & J	one-best	95.15	95.32	87.94	88.44
(2005)	reranked	95.81	96.04	88.64	89.17
Weighted intersection		96.32	96.47	89.32	89.80

Table 5: F-measure shallow bracketing accuracy when trained on WSJ sections 2-21 and applied to either WSJ section 00 or SWBD section 4. Systems include our shallow parser (SPRep); the Charniak and Johnson (2005) system (C & J), both initial one-best and reranked-best; and the weighted intersection between the reranked 50-best list and the SPRep system.

and Johnson (2005), which allows a comparison of the shallow parsing gains that they obtain from that system with those documented here. The reranker is a discriminatively trained Maximum Entropy model with an F-measure parsing accuracy objective. It uses a large number of features, and is applied to the 50-best output from the generative Charniak parsing model. The reranking model was trained on sections 2-21, with section 24 used as heldout. This allows us to compare its shallow parsing accuracy with other systems on the tasks that use this training setup: the Li & Roth task (testing on WSJ section 00) and the domain shift task (testing on Switchboard section 4). Table 5 shows two new trials making use of this reranking model.

The Charniak and Johnson (2005) system output (denoted *C & J* in the table) before reranking (denoted *one-best*) is identical to the Charniak (2000) results that have been reported in the other tables. After reranking (denoted *reranked*), the performance improves by roughly 0.7 percentage points for both tasks, nearly reaching the performance that we obtained with weighted intersection of the SPRep model and the n-best list before reranking. Weighted intersection between the reranked list and the shallow parser as described earlier, with a newly estimated scaling factor (α=30), provides a roughly 0.5 percentage point increase over the result obtained by the reranker. The difference between the Charniak output before and after reranking is statistically significant at $p < 0.001$, as is the difference between the reranked output and the weighted intersection, using the same test reported earlier.

3.5 Discussion

While it may be seen to be overkill to apply a context-free parser for these shallow parsing tasks, we feel that these results are very interesting for a couple of reasons. First, they go some way toward correcting the misperception that context-free parsers are less applicable in real-world scenarios than finite-state sequence models. Finite-state models are undeniably more efficient; however, it is important to have a clear idea of how much accuracy is being sacrificed to reach that efficiency. Any given application will need to examine the efficiency/accuracy trade-off with different objectives for optimality. For those willing to trade efficiency for accuracy, it is worthwhile knowing that it is possible to do much better on these tasks than what has been reported in the past.

4 Conclusion and Future Work

In summary, we have demonstrated in this paper that there is no accuracy or robustness benefit to shallow parsing with finite-state models over using high-accuracy context-free models. Even more, there is a large benefit to be had in combining the output of high-accuracy context-free parsers with the output of shallow parsers. We have demonstrated a large improvement over the previous-best reported results on several tasks, including the well-known NP-Chunking and CoNLL-2000 shallow parsing tasks.

Part of the misperception of the relative benefits of finite-state and context-free models is due to difficulty evaluating across these differing annotation styles. Mapping from context-free parser output to the shallow constituents defined in the CoNLL-2000 task depends on many construction-specific operations that have unfairly penalized context-free parsers in previous comparisons.

While the results of combining system outputs show one benefit of combining systems, as presented in this paper, they hardly exhaust the possibilities of exploiting the differences between these models. Making use of the scores for the shallow parses output by the Charniak parser is a demonstrably effective way to improve performance. Yet there are other possible features explicit in the context-free parse candidates, such as head-to-head dependencies, which might be exploited to further improve performance. We intend to explore including features from the context-free parser output in our perceptron model to improve shallow parsing accuracy.

Another possibility is to look at improving context-free parsing accuracy. Within a multi-pass parsing strategy, the high-accuracy shallow parses that result from system combination could be used to restrict the search within yet another pass of a context-free parser. That parser could then search for the best global analysis from within just the space of parses consistent with the provided shallow parse. Also, features of the sort used in our shallow parser could be included in a reranker, such as that in Charniak and Johnson (2005), with a context-free parsing accuracy objective.

A third possibility is to optimize the definition of the shallow-parse phrase types themselves, for use in other applications. The composition of the set of phrase types put forth by Sang and Buchholz (2000) may not be optimal for certain applications. One such application is discourse parsing, which relies on accurate detection of clausal boundaries. Shallow parsing could provide reliable information on the location of these boundaries, but the current set of phrase types may be too general for such use. For example, consider infinitival verb phrases, which often indicate the start of a clause whereas other types of verb phrases do not. Unfortunately, with only one VP category in the CoNLL-2000 set of phrase types, this distinction is lost. Expanding the defined set of phrase types could benefit many applications.

Future work will also include continued exploration of possible features that can be of use for either shallow parsing models or context-free parsing models. In addition, we intend to investigate ways in which to encode approximations to context-free parser derived features that can be used within finite-state models, thus perhaps preserving finite-state efficiency while capturing at least some of the accuracy gain that was observed in this paper.

Acknowledgments

We would like to thank Eugene Charniak and Mark Johnson for help with the parser and reranker documented in their paper. The first author of this paper was supported under an NSF Graduate Research Fellowship. In addition, this research was supported in part by NSF Grant #IIS-0447214. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Steven Abney. 1996. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4):337–344.

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4).

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of ACL*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of NAACL*, pages 132–139.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of ACL*, pages 16–23.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th ICML Conference*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on EMNLP*, pages 1–8.

L. Gillick and S. Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of ICASSP*, pages 532–535.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the 2nd Annual Meeting of NAACL*.

Xin Li and Dan Roth. 2001. Exploring evidence for shallow parsing. In *Proceedings of the 5th Conference on CoNLL*.

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.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 82–94.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 4th Conference on CoNLL*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the HLT-NAACL Annual Meeting*.

Tong Zhang, Fred Damerau, and David Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

Morphology and Reranking for the Statistical Parsing of Spanish

Brooke Cowan
MIT CSAIL
brooke@csail.mit.edu

Michael Collins
MIT CSAIL
mcollins@csail.mit.edu

Abstract

We present two methods for incorporating detailed features in a Spanish parser, building on a baseline model that is a lexicalized PCFG. The first method exploits Spanish morphology, and achieves an F1 constituency score of 83.6%. This is an improvement over 81.2% accuracy for the baseline, which makes little or no use of morphological information. The second model uses a reranking approach to add arbitrary global features of parse trees to the morphological model. The reranking model reaches 85.1% F1 accuracy on the Spanish parsing task. The resulting model for Spanish parsing combines an approach that specifically targets morphological information with an approach that makes use of general structural features.

1 Introduction

Initial methods for statistical parsing were mainly developed through experimentation on English data sets. Subsequent research has focused on applying these methods to other languages. There has been widespread evidence that new languages exhibit linguistic phenomena that pose considerable challenges to techniques originally developed for English; because of this, an important area of current research concerns how to model these phenomena more accurately within statistical approaches. In this paper, we investigate this question within the context of parsing Spanish. We describe two methods for incorporating detailed features in a Spanish parser, building on a baseline model that is a lexicalized PCFG originally developed for English.

Our first model uses morphology to improve the performance of the baseline model. English is a morphologically-impoverished language, while most of the world's languages exhibit far richer morphologies. Spanish is one of these languages. For instance, the forms of Spanish nouns, determiners, and adjectives reflect both number and gender; pronouns reflect gender, number, person, and case. Furthermore, morphological constraints may be manifested at the syntactic level: certain constituents of a noun phrase are constrained to agree in number and gender, and a verb is constrained to agree in number and person with its subject. Hence, morphology gives us important structural cues about how the words in a Spanish sentence relate to one another. The mechanism we employ for incorporating morphology into the PCFG model (the Model 1 parser in (Collins, 1999)) is the modification of its part-of-speech (POS) tagset; in this paper, we explain how this mechanism allows the parser to better capture morphological constraints.

All of the experiments in this paper are carried out using a freely-available Spanish treebank produced by the 3LB project (Navarro et al., 2003). This resource contains around 3,500 hand-annotated trees encoding ample morphological information. We could not use all of this information and adequately train the resulting parameters due to limited training data. Hence, we used development data to test the performance of several models, each incorporating a subset of morphological information. The highest-accuracy model on the development set uses the mode and number of verbs, as well as the number of adjectives, determiners, nouns, and pronouns. On test data, it reaches F1 accuracy of 83.6%/83.9%/79.4% for labeled constituents, unlabeled dependencies, and labeled dependencies, respectively. The baseline model, which makes almost no use of morphology, achieves 81.2%/82.5%/77.0% in these same measures.

We use the morphological model from the aforementioned experiments as a base parser in a second set of experiments. Here we investigate the efficacy of a reranking approach for parsing Spanish by using

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 795–802, Vancouver, October 2005. ©2005 Association for Computational Linguistics

arbitrary structural features. Previous work in statistical parsing (Collins and Koo, 2005) has shown that applying reranking techniques to the n-best output of a base parser can improve parsing performance. Applying an exponentiated gradient reranking algorithm (Bartlett et al., 2004) to the n-best output of our morphologically-informed Spanish parsing model gives us similar improvements. Using the reranking model combined with the morphological model raises performance to 85.1%/84.7%/80.2% F1 accuracy for labeled constituents, unlabeled dependencies, and labeled dependencies.

2 Related Work

The statistical parsing of English has surpassed 90% accuracy in the precision and recall of labeled constituents (e.g., (Collins, 1999; Charniak and Johnson, 2005)). A recent proliferation of treebanks in various languages has fueled research in the parsing of other languages. For instance, work has been done in Chinese using the Penn Chinese Treebank (Levy and Manning, 2003; Chiang and Bikel, 2002), in Czech using the Prague Dependency Treebank (Collins et al., 1999), in French using the French Treebank (Arun and Keller, 2005), in German using the Negra Treebank (Dubey, 2005; Dubey and Keller, 2003), and in Spanish using the UAM Spanish Treebank (Moreno et al., 2000). The best-reported F1 constituency scores from this work for each language are 79.9% (Chinese (Chiang and Bikel, 2002)), 81.0% (French (Arun and Keller, 2005), 76.2% (German (Dubey, 2005)), and 73.8% (Spanish (Moreno et al., 2000)). The authors in (Collins et al., 1999) describe an approach that gives 80% accuracy in recovering unlabeled dependencies in Czech.[1]

The project that is arguably most akin to the work presented in this paper is that on Spanish parsing (Moreno et al., 2000). However, a direct comparison of scores is complicated by the fact that we have used a different corpus as well as larger training and test sets (2,800- vs. 1,500-sentence training sets, and 700- vs. 40-sentence test sets).

[1]Note that cross-linguistic comparison of results is complicated: in addition to differences in corpus annotation schemes and sizes, there may be significant differences in linguistic characteristics.

Category	**Attributes**
Adjective	gender, number, participle
Determiner	gender, number, person, possessor
Noun	gender, number
Verb	gender, number, person, mode, tense
Preposition	gender, number, form
Pronoun	gender, number, person, case, possessor

Table 1: A list of the morphological features from which we created our models. For brevity, we only list attributes with at least two values. See (Civit, 2000) for a comprehensive list of the morphological attributes included in the Spanish treebank.

3 Models

This section details our two approaches for adding features to a baseline parsing model. First, we describe how morphological information can be added to a parsing model by modifying the POS tagset. Second, we describe an approach that reranks the n-best output of the morphologically-rich parser, using arbitrary, general features of the parse trees as additional information.

3.1 Adding Morphological Information

The mechanism we employ for incorporating morphological information is the modification of the POS tagset of a lexicalized PCFG[2] — the Model 1 parser described in (Collins, 1999) (hereafter Model 1). Each POS tagset can be thought of as a particular morphological model or a subset of morphological attributes. Table 1 shows the complete set of morphological features we considered for Spanish. There are 22 morphological features in total in this table; different POS sets can be created by deciding whether or not to include each of these 22 features; hence, there are 2^{22} different morphological models we could have created. For instance, one particular model might capture the modal information of verbs. In this model, there would be six POS tags for verbs (one for each of indicative, subjunctive, imperative, infinitive, gerund, and participle) instead of just one. A model that captured both the number and mode of verbs would have 18 verbal POS tags, assuming three values (singular, plural, and neutral) for the number feature.

The Effect of the Tagset on Model 1 Modifying the POS tagset allows Model 1 to better distinguish

[2]Hand-crafted head rules are used to lexicalize the trees.

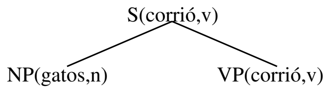


Figure 1: An ungrammatical dependency: the plural noun *gatos* is unlikely to modify the singular verb *corrió*.

events that are unlikely from those that are likely, on the basis of morphological evidence. An example will help to illustrate this point.

Model 1 relies on statistics conditioned on lexical headwords for practically all parameters in the model. This sensitivity to headwords is achieved by propagating lexical heads and POS tags to the nonterminals in the parse tree. Thus, any statistic based on headwords may also be sensitive to the associated POS tag. For instance, consider the subtree in Figure 1. Note that this structure is ungrammatical because the subject, *gatos* (*cats*), is plural, but the verb, *corrió* (*ran*), is singular. In Model 1, the probability of generating the noun phrase (NP) with headword *gatos* and headtag noun (n) is defined as follows:[3]

$$P(\text{gatos}, \mathsf{n}, \mathsf{NP} \mid \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) = P_1(\mathsf{n}, \mathsf{NP} \mid \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) \times P_2(\text{gatos} \mid \mathsf{n}, \mathsf{NP}, \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP})$$

The parser smooths parameter values using backed-off statistics, and in particular smooths statistics based on headwords with coarser statistics based on POS tags alone. This allows the parser to effectively use POS tags as a way of separating different lexical items into subsets or classes depending on their syntactic behavior. In our example, each term is estimated as follows:

$$\begin{aligned} &P_1(\mathsf{n}, \mathsf{NP} \mid \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) = \\ &\lambda_{1,1}\hat{P}_{1,1}(\mathsf{n}, \mathsf{NP} \mid \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) + \\ &\lambda_{1,2}\hat{P}_{1,2}(\mathsf{n}, \mathsf{NP} \mid \mathsf{v}, \mathsf{S}, \mathsf{VP}) + \\ &\lambda_{1,3}\hat{P}_{1,3}(\mathsf{n}, \mathsf{NP} \mid \mathsf{S}, \mathsf{VP}) \end{aligned}$$

and

$$\begin{aligned} &P_2(\text{gatos} \mid \mathsf{n}, \mathsf{NP}, \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) = \\ &\lambda_{2,1}\hat{P}_{2,1}(\text{gatos} \mid \mathsf{n}, \mathsf{NP}, \text{corrió}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) + \\ &\lambda_{2,2}\hat{P}_{2,2}(\text{gatos} \mid \mathsf{n}, \mathsf{NP}, \mathsf{v}, \mathsf{S}, \mathsf{VP}) + \\ &\lambda_{2,3}\hat{P}_{2,3}(\text{gatos} \mid \mathsf{n}) \end{aligned}$$

[3]Note that the parsing model includes other features such as distance which we omit from the parameter definition for the sake of brevity.

Here the $\hat{P}_{i,j}$ terms are maximum likelihood estimates derived directly from counts in the training data. The $\lambda_{i,j}$ parameters are defined so that $\lambda_{1,1}+\lambda_{1,2}+\lambda_{1,3} = \lambda_{2,1}+\lambda_{2,2}+\lambda_{2,3} = 1$. They control the relative contribution of each level of back-off to the final estimate.

Note that thus far our example has not included any morphological information in the POS tags. Because of this, we will see that there is a danger of the estimates P_1 and P_2 both being high, in spite of the dependency being ungrammatical. P_1 will be high because all three estimates $\hat{P}_{1,1}, \hat{P}_{1,2}$ and $\hat{P}_{1,3}$ will most likely be high. Next, consider P_2. Of the three estimates $\hat{P}_{2,1}, \hat{P}_{2,2}$, and $\hat{P}_{2,3}$, only $\hat{P}_{2,1}$ retains the information that the noun is plural and the verb is singular. Thus P_2 will be sensitive to the morphological clash between *gatos* and *corrió* only if $\lambda_{2,1}$ is high, reflecting a high level of confidence in the estimate of $\hat{P}_{2,3}$. This will only happen if the context $\langle$corrió, v, S, VP$\rangle$ is seen frequently enough for $\lambda_{2,1}$ to take a high value. This is unlikely, given that this context is quite specific. In summary, the impoverished model can only capture morphological restrictions through lexically-specific estimates based on extremely sparse statistics.

Now consider a model that incorporates morphological information — in particular, number information — in the noun and verb POS tags. *gatos* will have the POS pn, signifying a plural noun; *corrió* will have the POS sv, signifying a singular verb. All estimates in the previous equations will reflect these POS changes. For example, P_1 will now be estimated as follows:

$$\begin{aligned} &P_1(\mathsf{pn}, \mathsf{NP} \mid \text{corrió}, \mathsf{sv}, \mathsf{S}, \mathsf{VP}) = \\ &\lambda_{1,1}\hat{P}_{1,1}(\mathsf{pn}, \mathsf{NP} \mid \text{corrió}, \mathsf{sv}, \mathsf{S}, \mathsf{VP}) + \\ &\lambda_{1,2}\hat{P}_{1,2}(\mathsf{pn}, \mathsf{NP} \mid \mathsf{sv}, \mathsf{S}, \mathsf{VP}) + \\ &\lambda_{1,3}\hat{P}_{1,3}(\mathsf{pn}, \mathsf{NP} \mid \mathsf{S}, \mathsf{VP}) \end{aligned}$$

Note that the two estimates $\hat{P}_{1,1}$ and $\hat{P}_{1,2}$ include an (unlikely) dependency between the POS tags pn and sv. Both of these estimates will be 0, assuming that a plural noun is never seen as the subject of a singular verb. At the very least, the context $\langle$sv, S, VP$\rangle$ will be frequent enough for $\hat{P}_{1,2}$ to be a reliable estimate. The value for $\lambda_{1,2}$ will therefore be high, leading to a low estimate for P_1, thus correctly assigning low probability to the ungrammatical dependency. In summary, the morphologically-rich model can make use of non-lexical statistics such as $\hat{P}_{1,2}(\mathsf{pn}, \mathsf{NP} \mid \mathsf{sv}, \mathsf{S}, \mathsf{VP})$ which contain dependencies between POS tags and which will most likely be estimated reliably by the model.

3.2 The Reranking Model

In the reranking model, we use an n-best version of the morphologically-rich parser to generate a number of candidate parse trees for each sentence in training and test data. These parse trees are then represented through a combination of the log probability under the initial model, together with a large number of global features. A reranking model uses the information from these features to derive a new ranking of the n-best parses, with the hope of improving upon the baseline model. Previous approaches (e.g., (Collins and Koo, 2005)) have used a linear model to combine the log probability under a base parser with arbitrary features derived from parse trees. There are a variety of methods for training the parameters of the model. In this work, we use the algorithm described in (Bartlett et al., 2004), which applies the large-margin training criterion of support vector machines (Cortes and Vapnik, 1995) to the reranking problem.

The motivation for the reranking model is that a wide variety of features, which can essentially be sensitive to arbitrary context in the parse trees, can be incorporated into the model. In our work, we included all features described in (Collins and Koo, 2005). As far as we are aware, this is the first time that a reranking model has been applied to parsing a language other than English. One goal was to investigate whether the improvements seen on English parsing can be carried across to another language. We have found that features in (Collins and Koo, 2005), initially developed for English parsing, also give appreciable gains in accuracy when applied to Spanish.

4 Data

The Spanish 3LB treebank is a freely-available resource with about 3,500 sentence/tree pairs that we have used to train our models. The average sentence length is 28 tokens. The data is taken from 38 complete articles and short texts. Roughly 27% of the texts are news articles, 27% scientific articles, 14% narrative, 11% commentary, 11% sports articles, 6% essays, and 5% articles from weekly magazines. The trees contain information about both constituency structure and syntactic functions.

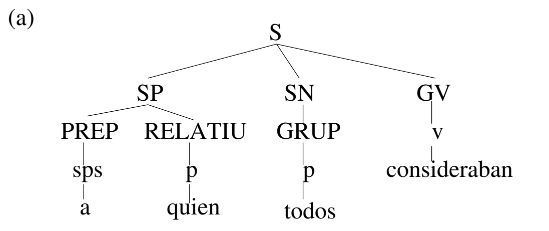
Non-Terminal	Significance
aq	*adjective*
cc	*conjunction*
COORD	*coordinated phrase*
ESPEC	*determiner*
GRUP	*base noun phrase*
GV	*verb phrase*
MORF	*impersonal pronoun*
p	*pronoun*
PREP	*base prepositional phrase*
RELATIU	*relative pronoun phrase*
s	*adjectival phrase*
SN	*noun phrase*
SP	*prepositional phrase*
SADV	*adverbial phrase*
S	*sentence*
sps	*preposition*
v	*verb*

Table 2: The non-terminals and preterminals from the Spanish 3LB corpus used in this paper.

4.1 Preprocessing

It is well-known that tree representation influences parsing performance (Johnson, 1998). Prior to training our models, we made some systematic modifications to the corpus trees in an effort to make it easier for Model 1 to represent the linguistic phenomena present in the trees. For the convenience of the reader, Table 2 gives a key to the non-terminal labels in the 3LB treebank that are used in this section and the remainder of the paper.

Relative and Subordinate Clauses Cases of relative and subordinate clauses appearing in the corpus trees have the basic structure of the example in Figure 2a. Figure 2b shows the modifications we impose on such structures. The modified structure has the advantage that the SBAR selects the CP node as its head, making the relative pronoun *que* the headword for the root of the subtree. This change allows, for example, better modeling of verbs that select for particular complementizers. In addition, the new subtree rooted at the S node now looks like a top-level sentence, making sentence types more uniform in structure and easier to model statistically. Additionally, the new structure differentiates phrases em-



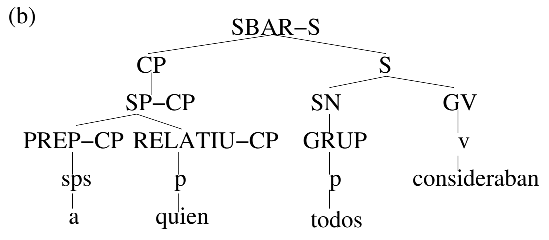
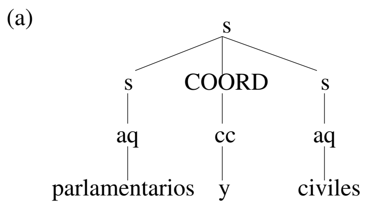


Figure 2: Figure (a) is the original structure from the 3LB treebank for the phrase *a quien todos consideraban* or *whom everyone considered*. We transform structures like (a) into (b) by inserting SBAR and CP nodes, and by marking all non-terminals below the CP with a -CP tag.



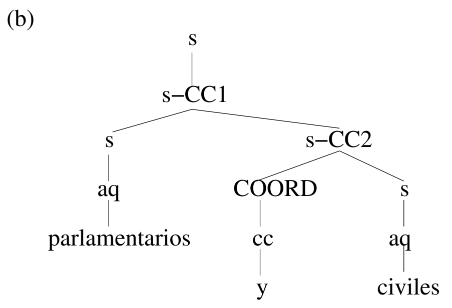


Figure 3: In the 3LB corpus, phrases involving coordination, are represented with a flat structure as in (a). For coordination involving a non-terminal X ($X = \texttt{s}$ in the example), we insert new nodes X-CC1 and X-CC2 to form the structure in (b).

bedded in the complementizers of SBARs from those used in other contexts, allowing relative pronouns like *quien* in Figure 2 to surface as lexical headwords when embedded in larger phrases beneath the CP node.[4]

Coordination In the treebank, coordinated constituents and their coordinating conjunction are placed as sister nodes in a flat structure. We enhance the structure of such subtrees, as in Figure 3. Our structure helps to rule out unlikely phrases such as *cats and dogs and*; the model trained with the original treebank structures will assign non-zero probability to ill-formed structures such as these.

5 Experiments

Our models were trained using a training set consisting of 80% of the data (2,801 sentence/tree pairs, 75,372 words) available to us in the 3LB treebank. We reserved the remaining 20% (692 sentences, 19,343 words) to use as unseen data in a test set. We selected these subsets with two criteria in mind: first, respecting the boundaries of the texts by placing articles in their entirety into either one subset or the other; and second, maintaining, in each subset, the same proportion of genres found in the original set of trees. During development, we used a cross-validation approach on the training set to test different models. We divided the 2,800 training data trees into 14 different development data sets, where each of these data sets consisted of 2,600 training sentences and 200 development sentences. We took the average over the results of the 14 splits to gauge the effectiveness of the model being tested.

To evaluate our models, we considered the recovery of labeled and unlabeled dependencies as well as labeled constituents. Unlabeled dependencies capture how the words in a sentence depend on one another. Formally, they are tuples $\{$*headchild index*, *modifier index*$\}$, where the indices indicate position in the sentence. Labeled dependencies include the labels of the modifier, headchild, and parent non-terminals as well. The root of the tree has a special dependency: $\{$*head index*$\}$ in the unlabeled case and $\{$*TOP*, *headchild index*, *root non-terminal*$\}$ in the labeled case. The labeled constituents in a tree are all of the non-terminals and, for each, the positions of the words it spans. We use the standard definitions of precision, recall, and F-measure.[5]

[4]This is achieved through our head rules.

[5]When extracting dependencies, we replaced all non-punctuation POS labels with a generic label *TAG* to avoid conflating tagging errors with dependency errors. We also included the structural changes that we imposed during preprocessing. Results for constituent precision and recall were computed after we restored the trees to the original treebank structure.

		Labeled Dep		Unlabeled Dep		Labeled Const			
						<=70 words		<=40 Words	
	Model	**Prec/Rec**	**Gain**	**Prec/Rec**	**Gain**	**Prec**	**Rec**	**Prec**	**Rec**
1	Baseline	76.0	—	82.1	—	81.6	80.4	82.6	81.4
2	n(P,N,V)	78.4	2.4	83.6	1.5	83.1	82.5	84.1	83.4
3	n(A,D,N,P,V)	78.2	2.2	83.5	1.4	83.3	82.4	84.2	83.3
4	n(V)	77.8	1.8	82.9	0.8	82.3	81.6	83.1	82.2
5	m(V)	78.4	2.4	83.1	1.0	82.8	82.0	83.8	82.9
6	t(V)	77.6	1.6	82.7	0.6	82.4	81.4	83.2	82.3
7	p(V)	78.1	2.1	83.3	1.2	82.9	82.0	83.8	82.8
8	g(V)	76.3	0.3	82.2	0.1	81.6	80.6	82.7	81.7
9	n(A,D,N,V,P)+m(V)	**79.0**	**3.0**	**84.0**	**1.9**	**83.9**	**83.2**	**84.7**	**84.1**
10	n(P,N,V)+m(V)	78.9	2.9	83.7/83.8	1.6/1.7	83.6	82.8	84.6	83.7
11	n(A,D,N,V,P)+m(V)+p(V)	78.7	2.7	83.6	1.5	83.6	82.9	84.4	83.8
12	n(A,D,N,V,P)+p(V)	78.4	2.4	83.5/83.6	1.4/1.5	83.3	82.6	84.2	83.5
13	n(A,D,N,V,P)+g(A,D,N,V,P)	78.1	2.1	83.2	1.1	83.1	82.5	83.9	83.4

Table 3: Results after training morphological models during development. When precision and recall differ in labeled or unlabeled dependencies, both scores are shown. Row 1 shows results on a baseline model containing almost no morphological information. The subsequent rows represent a subset of the models with which we experimented: n(P,N,V) uses number for pronouns, nouns, and verbs; n(A,D,N,P,V) uses number for adjectives, determiners, nouns, pronouns, and verbs; n(V) uses number for verbs; m(V) uses mode for verbs; t(V) uses tense for verbs; p(V) uses person for verbs; g(V) uses gender for verbs; the models in rows 9–12 are combinations of these models, and in row 13, n(A,D,N,V,P) combines with g(A,D,N,V,P), which uses gender for adjectives, determiners, nouns, verbs, and pronouns. The results of the best-performing model are in bold.

		Labeled Dep	Unlabeled Dep	Labeled Const			
				<=70 words		<=40 Words	
	Model	**Prec/Rec**	**Prec/Rec**	**Prec**	**Rec**	**Prec**	**Rec**
1	Baseline	77.0	82.5	81.7	80.8	83.1	82.0
2	n(A,D,N,V,P)+m(V)	79.4	83.9	83.9	83.4	85.1	84.4
3	RERANK	80.2	84.7	85.2	85.0	86.3	85.9

Table 4: Results after running the morphological and reranking models on test data. Row 1 is our baseline model. Row 2 is the morphological model that scored highest during development. Row 3 gives the accuracy of the reranking approach, when applied to n-best output from the model in Row 2.

5.1 The Effects of Morphology

In our first experiments, we trained over 50 models, incorporating different morphological information into each in the way described in Section 3.1. Prior to running the parsers, we trained the POS tagger described in (Collins, 2002). The output from the tagger was used to assign a POS label for unknown words. We only attempted to parse sentences under 70 words in length.

Table 3 describes some of the models we tried during development and gives results for each. Our baseline model, which we used to evaluate the effects of using morphology, was Model 1 (Collins, 1999) with a simple POS tagset containing almost no morphological information. The morphological models we show are meant to be representative of both the highest-scoring models and the performance of various morphological features. For instance, we found that, in general, gender had only a slight impact on the performance of the parser. Note that gender is not a morphological attribute of Spanish verbs, and that the inclusion of verbal features, particularly number, mode, and person, generated the strongest-performing models in our experiments.

Table 4 shows the results of running two models on the test set: the baseline model and the best-performing morphological model from the development stage. This model uses the number and mode of verbs, as well as the number of adjectives, determiners, nouns, and pronouns.

The results in Tables 3 and 4 show that adding some amount of morphological information to a parsing model is beneficial. We found, however, that adding more information does not always lead to improved performance (see, for example, rows 11 and 13 in Table 3). Presumably this is because the tagset grows too large.

Table 5 takes a closer look at the performance of the best-performing morphological model in the recovery of particular labeled dependencies. The breakdown shows the top 15 dependencies in the gold-standard trees across the entire training set. Collectively, these dependencies represent around 72% of the dependencies seen in this data.

We see an extraordinary gain in the recovery of some of these dependencies when we add morphological information. Among these are the two involving postmodifiers to verbs. When examining the output of the morphological model, we found that much of this gain is due to the fact that there are two non-terminal labels used in the treebank that specify modal information of verbs they dominate (infinitivals and gerunds): with insufficient morphological information, the baseline parser was unable to distinguish regular verb phrases from these more specific verb phrases.

Some dependencies are particularly difficult for the parser, such as that in which `SBAR` modifies a noun ({`GRUP TAG SBAR` R}). We found that around 20% of cases of this type in the training set involve structures like *el proceso de negociones que* (in English *the process of negotiation that*). This type of structure is inherently difficult to disambiguate. In Spanish, such structures may be more common than in English, since phrases involving nominal modifiers to nouns, like *negotiation process*, are always formed as *noun* + *de* + *noun*.

Dependency	Count	Model	Prec/Rec
Determiner modifier	9680	BL	95.0/95.4
`SN GRUP ESPEC` L	(15.5%)	M	95.4/95.7
Complement of `SP`	9052	BL	92.4/92.9
`SP PREP SN` R	(14.5%)	M	93.2/93.9
`SP` modifier to noun	4500	BL	83.9/78.1
`GRUP TAG SP` R	(7.2%)	M	82.9/79.9
Subject	3106	BL	77.7/86.1
`S GV SN` L	(5.0%)	M	83.1/87.5
Sentential head	2758	BL	75.0/75.0
`TOP S`	(4.4%)	M	79.7/79.7
`S` modifier under `SBAR`	2728	BL	83.3/82.1
`SBAR CP S` R	(4.4%)	M	86.0/84.7
`SP` modifier to verb	2685	BL	62.4/78.8
`S GV SP` R	(4.3%)	M	72.6/82.5
`SN` modifier to verb	2677	BL	71.6/75.6
`S GV SN` R	(4.3%)	M	81.0/83.0
Adjective postmodifier	2522	BL	76.3/83.6
`GRUP TAG s` R	(4.0%)	M	76.4/83.5
Adjective premodifier	980	BL	79.2/80.0
`GRUP TAG s` L	(1.6%)	M	80.1/79.3
`SBAR` modifier to noun	928	BL	62.2/60.6
`GRUP TAG SBAR` R	(1.4%)	M	61.3/60.8
Coordination	895	BL	65.2/72.7
`S-CC2 S coord` L	(1.4%)	M	66.7/74.2
Coordination	870	BL	52.4/56.1
`S-CC1 S-CC2 S` L	(1.4%)	M	60.3/63.6
Impersonal pronoun	804	BL	93.3/96.4
`S GV MORF` L	(1.3%)	M	92.0/95.6
`SN` modifier to noun	736	BL	47.3/39.5
`GRUP TAG SN` R	(1.2%)	M	51.7/50.8

Table 5: Labeled dependency accuracy for the top 15 dependencies (representing around 72% of all dependencies) in the gold-standard trees across all training data. The first column shows the type and subtype, where the subtype is specified as the 4-tuple {*parent non-terminal*, *head non-terminal*, *modifier non-terminal*, *direction*}; the second column shows the count for that subtype and the percent of the total that it represents (where the total is 62,372) . The model BL is the baseline, and M is the morphological model n(A,D,N,V,P)+m(V).

5.2 Experiments with Reranking

In the reranking experiments, we follow the procedure described in (Collins and Koo, 2005) for creation of a training set with n-best parses for each sentence. This method involves jack-knifing the data: the training set of 2,800 sentences was parsed in 200-sentence chunks by an n-best morphological parser trained on the remaining 2,600 sentences. This ensured that each sentence in the training data had n-best output from a baseline model that was not trained on that sentence. We used the optimal morphological model (n(A,D,N,V,P)+m(V)) to generate the n-best lists, and we used the feature set described in (Collins and Koo, 2005). The test results are given in Table 4.[6]

5.3 Statistical Significance

We tested the significance of the labeled precision and recall results in Table 4 using the sign test. When applying the sign test, for each sentence in the test data we calculate the sentence-level F1 constituent score for the two parses being compared. This indicates whether one model performs better on that sentence than the other model, or whether the two models perform equally well, information used by the sign test. All differences were found to be statistically significant at the level $p = 0.01$.[7]

[6]Note that we also created development sets for development of the reranking approach, and for cross-validation of the single parameter C in approach of (Bartlett et al., 2004).

[7]When comparing the baseline model to the morphological model on the 692 test sentences, F1 scores improved on 314 sentences, and became worse on 164 sentences. When comparing the baseline model to the reranked model, 358/157 sen-

6 Conclusions and Future Work

We have developed a statistical parsing model for Spanish that performs at 85.1% F1 constituency accuracy. We find that an approach that explicitly represents some of the particular features of Spanish (i.e., its morphology) does indeed help in parsing. Moreover, this approach is compatible with the reranking approach, which uses general features that were first developed for use in an English parser. In fact, our best parsing model combines both the language-specific morphological features and the non-specific reranking features. The morphological features are local, being restricted to dependencies between words in the parse tree; the reranking features are more global, relying on larger portions of parse structures. Thus, we see our final model as combining the strengths of two complementary approaches.

We are curious to know the extent to which a close analysis of the dependency errors made by the baseline parser can be corrected by the development of features tailored to addressing these problems. Some preliminary investigation of this suggests that we see much higher gains when using generic features than these more specific ones, but we leave a thorough investigation of this to future work. Another avenue for future investigation is to try using a more sophisticated baseline model such as Collins' Model 2, which incorporates both subcategorization and complement/adjunct information. Finally, we would like to use the Spanish parser in an application such as machine translation.

Acknowledgements

We would like to thank Xavier Carreras for pointing us to the Spanish 3LB treebank and Montserrat Civit for providing access to the data and answering questions about it. We also gratefully acknowledge the support of the National Science Foundation under grants 0347631 and 0434222.

tences had improved/worse parses. When comparing the morphological model to the reranked model, 199/106 sentences had improved/worse parses.

References

Abhishek Arun and Frank Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: the case of French. ACL 2005, Ann Arbor, MI.

Peter Bartlett, Michael Collins, Ben Taskar, and David McAllester. 2004. Exponentiated gradient algorithms for large-margin structured classification. *Proceedings of NIPS 2004*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. ACL 2005, Ann Arbor, MI.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. *Proceedings of COLING-2002*, pages 183–189.

Montserrat Civit Torruella. 2000. Guía para la anotación morfosintáctica del corpus CLiC-TALP. X-Tract Working Paper, WP-00/06.

Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. University of Pennsylvania.

Michael Collins, Jan Hajic, Lance Ranshaw, and Christoph Tillman. 1999. A statistical parser for Czech. ACL 99.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. EMNLP 2002.

Michael Collins and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–69.

C. Cortes and V. Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273–297.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. ACL 2003, pp. 96–103.

Amit Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. ACL 2005, Ann Arbor, MI.

Mark Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632.

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? ACL 2003, pp. 439–446.

Antonio Moreno, Ralph Grishman, Susana López, Fernando Sánchez, and Satoshi Sekine. 2000. A treebank of Spanish and its application to parsing. *The Proceedings of the Workshop on Using Evaluation within HLT Programs: Results and Trends*, Athens, Greece.

Borja Navarro, Montserrat Civit, Ma. Antònia Martí, Raquel Marcos, and Belén Fernández. 2003. Syntactic, semantic and pragmatic annotation in Cast3LB. *Shallow Processing of Large Corpora* (SProLaC), a Workshop of Corpus Linguistics, 2003, Lancaster, UK.

Some Computational Complexity Results for Synchronous Context-Free Grammars

Giorgio Satta
Dept. of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova
Italy
satta@dei.unipd.it

Enoch Peserico
Dept. of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova
Italy
enoch@dei.unipd.it

Abstract

This paper investigates some computational problems associated with probabilistic translation models that have recently been adopted in the literature on machine translation. These models can be viewed as pairs of probabilistic context-free grammars working in a 'synchronous' way. Two hardness results for the class NP are reported, along with an exponential time lower-bound for certain classes of algorithms that are currently used in the literature.

1 Introduction

State of the art architectures for machine translation are all based on mathematical models called translation models. Generally speaking, a translation model accounts for all the elementary operations that rule the process of translation between the words and the different word orderings of the source and target languages. Translation models are usually enriched with statistical parameters, to drive the search toward the most likely translation(s). Specialized algorithms are provided for the automatic estimation of these parameters from corpora of translation pairs. Besides the task of natural language translation, statistical translation models are also exploited in other applications, such as word alignment, multilingual document retrieval and automatic dictionary construction.

The most successful translation models that are found in the literature exploit finite-state machinery. The approach started with the so-called IBM models (Brown et al., 1988), implementing a set of elementary operations, such as movement, duplication and translation, that independently act on individual words in the source sentence. These word-to-word models have been later enriched with the introduction of larger units such as phrases; see for instance (Och et al., 1999; Och and Ney, 2002). Still, the generative capacity of these models lies within the realm of finite-state machinery (Kumar and Byrne, 2003), so they are unable to handle nested structures and do not provide the expressivity required to process language pairs with very different word orderings.

Recently, more sophisticated translation models have been proposed, borrowing from the theory of compilers and making use of synchronous rewriting. In synchronous rewriting, two formal grammars are exploited, one describing the source language and the other describing the target language. Furthermore, the productions of the two grammars are paired and, in the rewriting process, such pairs are always applied synchronously. Formalisms based on synchronous rewriting have been empowered with the use of statistical parameters, and specialized estimation and translation (decoding) algorithms were newly developed. Among the several proposals, we mention here the models presented in (Wu, 1997; Wu and Wong, 1998), (Alshawi et al., 2000), (Yamada and Knight, 2001), (Gildea, 2003) and (Melamed, 2003).

In this paper we consider synchronous models based on context-free grammars and probabilistic extensions thereof. This is the most common choice in statistical translation models that exceed the generative power of finite-state machinery. We focus on two associated computational problems that have been defined in the literature. One is the membership problem, which involves testing whether an input string pair can be generated by the model. The other is the translation problem (also called the decoding problem) which involves the search for a suitable translation of an input string/structure. It has been often informally stated in the literature that the use of structured models results in efficient, polynomial time algorithms for the above problems. We show here that sometimes this is not the case. The contribution of this paper can be stated as follows:

- we show that the membership problem is NP-hard, unless a constant bound is imposed on the length of the productions (Section 3);
- we show an exponential time lower bound for the membership problem, in case chart parsing is adopted (Section 3);
- we show that translating an input string into the best parse tree in the target language is NP-hard, even in case productions are bounded in length (Section 4).

Investigation of the computational complexity of translation models has started in (Knight, 1999) for word-to-word models. This paper can be seen as the continuation of that line of research.

2 Synchronous context-free grammars

Several definitions for synchronous context-free grammars have been proposed in the literature; see for instance (Chiang, 2004; Chiang, 2005). Our definition is based on syntax-directed translation schemata (SDTS; Aho and Ullman, 1972), with the difference that we do not impose the restriction that two paired context-free productions have the same left-hand side. As it will be discussed in Section 4, this results in an enriched generative capacity when probabilistic extensions are considered. We assume the reader is familiar with the definition of context-free grammar (CFG) and with the associated notion of derivation.

Let V_N and V_T be sets of nonterminal and terminal symbols, respectively. In what follows we need to represent bijections between all the occurrences of nonterminals in two strings over $V_N \cup V_T$. This can be done by annotating nonterminals with indices from an infinite set. We define $\mathcal{I}(V_N) = \{A^{(t)} \mid A \in V_N,\ t \in \mathbb{N}\}$ and $V_I = \mathcal{I}(V_N) \cup V_T$. We write $\mathsf{index}(\gamma)$, $\gamma \in V_I^*$, to denote the set of all indices (the integers t) that appear in symbols in γ. Two strings $\gamma, \gamma' \in V_I^*$ are **synchronous** if each index in $\mathsf{index}(\gamma)$ occurs only once in γ, each index in $\mathsf{index}(\gamma')$ occurs only once in γ', and $\mathsf{index}(\gamma) = \mathsf{index}(\gamma')$. Therefore synchronous strings have the general form

$$\begin{array}{c} u_{10}A_{11}^{(t_1)}u_{11}A_{12}^{(t_2)}u_{12} \quad \cdots \quad u_{1r-1}A_{1r}^{(t_r)}u_{1r}, \\ u_{20}A_{21}^{(t_{\pi(1)})}u_{21}A_{22}^{(t_{\pi(2)})}u_{22} \quad \cdots \quad u_{2r-1}A_{2r}^{(t_{\pi(r)})}u_{2r}, \end{array}$$

where $r \geq 0$, $u_{1i}, u_{2i} \in V_T^*$, $A_{1i}^{(t_i)}, A_{2i}^{(t_{\pi(i)})} \in \mathcal{I}(V_N)$, $t_i \neq t_j$ for $i \neq j$ and π is some permutation defined on set $\{1, \ldots, r\}$.

Definition 1 *A* **synchronous context-free grammar** *(SCFG) is a tuple $G = (V_N, V_T, P, S)$, where V_N, V_T are finite, disjoint sets of nonterminal and terminal symbols, respectively, $S \in V_N$ is the start symbol and P is a finite set of synchronous productions, each of the form $[A_1 \rightarrow \alpha_1,\ \ A_2 \rightarrow \alpha_2]$, with $A_1, A_2 \in V_N$ and $\alpha_1, \alpha_2 \in V_I^*$ synchronous strings.*

The size of a SCFG G is defined as $|G| = \sum_{[A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2] \in P} |A_1\alpha_1A_2\alpha_2|$. Based on an example from (Yamada and Knight, 2001), we provide a sample SCFG fragment translating from English to Japanese, specified by means of the following synchronous productions:

$$\begin{array}{ll} s_1 : & [\mathrm{VB} \rightarrow \mathrm{PRP}^{(1)}\ \mathrm{VB1}^{(2)}\ \mathrm{VB2}^{(3)}, \\ & \ \mathrm{VB} \rightarrow \mathrm{PRP}^{(1)}\ \mathrm{VB2}^{(3)}\ \mathrm{VB1}^{(1)}] \\ s_2 : & [\mathrm{VB2} \rightarrow \mathrm{VB}^{(1)}\ \mathrm{TO}^{(2)}, \\ & \ \mathrm{VB2} \rightarrow \mathrm{TO}^{(2)}\ \mathrm{VB}^{(1)}\ \mathrm{ga}] \\ s_3 : & [\mathrm{TO} \rightarrow \mathrm{TO}^{(1)}\ \mathrm{NN}^{(2)},\ \ \mathrm{TO} \rightarrow \mathrm{NN}^{(2)}\ \mathrm{TO}^{(1)}] \\ s_4 : & [\mathrm{PRP} \rightarrow \mathrm{he},\ \ \mathrm{PRP} \rightarrow \mathrm{kare\ ha}] \\ s_5 : & [\mathrm{VB1} \rightarrow \mathrm{adores},\ \ \mathrm{VB1} \rightarrow \mathrm{daisuki\ desu}] \\ s_6 : & [\mathrm{VB} \rightarrow \mathrm{listening},\ \ \mathrm{VB} \rightarrow \mathrm{kiku\ no}] \\ s_7 : & [\mathrm{TO} \rightarrow \mathrm{to},\ \ \mathrm{TO} \rightarrow \mathrm{wo}] \\ s_8 : & [\mathrm{NN} \rightarrow \mathrm{music},\ \ \mathrm{NN} \rightarrow \mathrm{ongaku}] \end{array}$$

Note that in production s_2 above, the nonterminals VB and TO generated from nonterminal VB2 in the English component are inverted in the Japanese component, where some additional lexical material is also added.

In a SCFG, the 'derives' relation is defined on synchronous strings in terms of simultaneous rewriting of two nonterminals with the same index. Some additional notation will help us defining this relation precisely. A **reindexing** is a one-to-one function on $\mathbb{N}$. We extend a reindexing f to V_I by letting $f(A^{(t)}) = A^{(f(t))}$ for $A^{(t)} \in \mathcal{I}(V_N)$ and $f(a) = a$ for $a \in V_T$. We also extend f to strings in V_I^* by letting $f(\varepsilon) = \varepsilon$ and $f(X\gamma) = f(X)f(\gamma)$, for each $X \in V_I$ and $\gamma \in V_I^*$. We say that strings $\gamma_1, \gamma_2 \in V_I^*$ are **independent** if $\mathsf{index}(\gamma_1) \cap \mathsf{index}(\gamma_2) = \emptyset$.

Definition 2 *Let $G = (V_N, V_T, P, S)$ be a SCFG and let γ_1, γ_2 be synchronous strings in V_I^*. The* **derives** *relation $[\gamma_1,\ \gamma_2] \Rightarrow_G [\delta_1,\ \delta_2]$ holds whenever there exist an index t in $\mathsf{index}(\gamma_1)$, a synchronous production $[A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2]$ in P and some reindexing f such that*

(i) *$f(\alpha_1\alpha_2)$ and $\gamma_1\gamma_2$ are independent; and*

(ii) *$\gamma_i = \gamma_i' A_i^{(t)} \gamma_i''$, $\delta_i = \gamma_i' f(\alpha_i) \gamma_i''$, for $i = 1, 2$.*

We also write $[\gamma_1,\ \gamma_2] \Rightarrow_G^s [\delta_1,\ \delta_2]$ to explicitly indicate that the derives relation holds through some synchronous production $s \in P$.

Since δ_1 and δ_2 in Definition 2 are synchronous strings, we can define the reflexive and transitive closure of $\Rightarrow_G$, written $\Rightarrow_G^*$. This relation is used to represent derivations in G. In case we have $[\gamma_{1i-1},\ \gamma_{2i-1}] \Rightarrow_G^{s_i} [\gamma_{1i},\ \gamma_{2i}]$ for $1 \leq i \leq n$, $n \geq 1$, we also write $[\gamma_{10},\ \gamma_{20}] \Rightarrow_G^{\sigma} [\gamma_{1n},\ \gamma_{2n}]$, where $\sigma = s_1 s_2 \cdots s_n$. We always assume some canonical form for derivations (as for instance leftmost derivation on the left component). Similarly to the case of context-free grammars, each derivation in G can be associated with a pair of parse trees, that is, one parse tree for each dimension.

Back to our example, we report a fragment of a derivation of the string pair [he adores listening to music, kare ha ongaku wo kiku no ga daisuki desu]:

$$\begin{aligned}
&[\mathrm{VB}^{(1)},\ \mathrm{VB}^{(1)}] \\
&\Rightarrow_G^{s_1} [\mathrm{PRP}^{(2)}\ \mathrm{VB1}^{(3)}\ \mathrm{VB2}^{(4)}, \\
&\qquad \mathrm{PRP}^{(2)}\ \mathrm{VB2}^{(4)}\ \mathrm{VB1}^{(3)}] \\
&\Rightarrow_G^{s_4} [\text{he } \mathrm{VB1}^{(3)}\ \mathrm{VB2}^{(4)}, \\
&\qquad \text{kare ha } \mathrm{VB2}^{(4)}\ \mathrm{VB1}^{(3)}] \\
&\Rightarrow_G^{s_5} [\text{he adores } \mathrm{VB2}^{(4)}, \\
&\qquad \text{kare ha } \mathrm{VB2}^{(4)} \text{ daisuki desu}] \\
&\Rightarrow_G^{s_2} [\text{he adores } \mathrm{VB}^{(5)}\ \mathrm{TO}^{(6)}, \\
&\qquad \text{kare ha } \mathrm{TO}^{(6)}\ \mathrm{VB}^{(5)} \text{ ga daisuki desu}].
\end{aligned}$$

The **translation** generated by a SCFG G is a binary relation over V_T^* defined as

$$T(G) \;=\; \{[w_1,\ w_2] \mid [S^{(1)},\ S^{(1)}] \Rightarrow_G^* [w_1,\ w_2],\ w_1, w_2 \in V_T^*\}.$$

The set of strings that are translations of a given string w_1 is defined as:

$$T(G, w_1) \;=\; \{w_2 \mid [w_1,\ w_2] \in T(G)\}.$$

A **probabilistic** SCFG (PSCFG) is a pair (G, p_G) where $G = (V_N, V_T, P, S)$ is a SCFG and p_G is a function from P to real numbers in $[0, 1]$ such that, for each $A_1, A_2 \in V_N$, we have:

$$\sum_{\alpha_1, \alpha_2} p_G([A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2] \;=\; 1.$$

If for $n \geq 1$ and $s_i \in P$, $1 \leq i \leq n$, string $\sigma = s_1 s_2 \cdots s_n$ is a canonical derivation of the form $[S^{(1)},\ S^{(1)}] \Rightarrow_G^{\sigma} [w_1,\ w_2]$, we write $p_G(\sigma) = \prod_{i=1}^{n} p_G(s_i)$. If $D([w_1, w_2])$ is the set of all canonical derivations in G for pair $[w_1, w_2]$, we write $p_G([w_1, w_2]) = \sum_{\sigma \in D([w_1, w_2])} p_G(\sigma)$.

3 The membership problem

We consider here the **membership** problem for SCFG, defined as follows: for input instance a SCFG G and a pair $[w_1,\ w_2]$, decide whether $[w_1,\ w_2]$ is in $T(G)$. This problem has been considered for instance in (Wu, 1997) for his inversion transduction grammars and has applications in the support of several tasks of automatic annotation of parallel corpora, as for instance segmentation, bracketing, phrasal and word alignment. We show that the membership problem for SCFGs is NP-hard. The result could be derived from the findings in (Melamed et al., 2004) that synchronous rewriting systems as SCFGs are related to the class of so called linear context-free rewriting systems (LCFRSs) and from the result that the membership problem for

LCFRSs is NP-hard (Satta, 1992; Kaji and others, 1994). However, we provide here a direct proof, to simplify the presentation.

Theorem 1 *The membership problem for SCFGs is NP-hard.*

Proof. We reduce from the three-satisfiability problem (3SAT, Garey and Johnson, 1979). Let $\langle U, C\rangle$ be an instance of the 3SAT problem, where $U = \{u_1, \ldots, u_p\}$ is a set of variables and $C = \{c_1, \ldots, c_n\}$ is a set of clauses. Each clause is a set of three literals from $\{u_1, \overline{u}_1, \ldots, u_p, \overline{u}_p\}$.

The general idea of the proof is to use a string pair $[w_1 w_2 \cdots w_p,\ w_c]$, where w_c is a string representation of C and each w_i is a string controlling the truth assignment for the variable u_i. We then construct a SCFG G such that each w_i can be derived in two possible ways only, using some specialized productions of G, encoding the truth assignment of variable u_i. In this way the derivation of the whole string $w_1 \cdots w_p$ in the left dimension corresponds to a guess of a truth assignment for U. Accordingly, on the right dimension only those symbols of w_c will be derived that represent clauses that hold true under the guessed assignment.

We need some additional notation. Below we treat C as an alphabet of atomic symbols. We use a function d such that, for every i with $1 \leq i \leq p$, $c_{d(i,1)}, c_{d(i,2)}, \ldots, c_{d(i,s_i)}$ is the sequence of all clauses that include literal u_i, in the left to right order in which they appear within $c_1 c_2 \cdots c_n$, and $c_{d(i,s_i+1)}, c_{d(i,s_i+2)}, \ldots, c_{d(i,t_i)}$ is the sequence of all clauses that include literal $\overline{u}_i$, again as they appear within $c_1 c_2 \cdots c_n$ from left to right. Note that we must have $\sum_{i=1}^{p} t_i = 3n$. We also use a function e such that, for every $1 \leq i \leq p$ and $1 \leq j \leq t_i$, $e(i,j) = j + \sum_{k=1}^{i-1} t_k$ (assume $\sum_{k=1}^{0} t_k = 0$).

Consider the alphabet $\{a_i, b_i \mid 1 \leq i \leq p\}$. For every i, $1 \leq i \leq p$, let w_i denote a sequence of exactly $t_i + 1$ alternating symbols a_i and b_i, i.e., $w_i \in (a_i b_i)^+ \cup (a_i b_i)^* a_i$. For every $1 \leq i \leq p$, let $x(i,1) = a_i b_i$ and let $x(i,h) = a_i$ (resp. b_i) if h is even (resp. odd), $2 \leq h \leq t_i$. Let also $\overline{x}(i,h) = a_i$ (resp. b_i) if h is odd (resp. even), $1 \leq h \leq t_i - 1$, and let $\overline{x}(i,t_i) = a_i b_i$ (resp. $b_i a_i$) if t_i is odd (resp. even). Therefore we can write $w_i = x(i,1)x(i,2)\cdots x(i,t_1) = \overline{x}(i,1)\overline{x}(i,2)\cdots\overline{x}(i,t_1)$.

Finally, we need a permutation π defined on the set $\{1, \ldots, 3n\}$ as follows. Fix i and j with $1 \leq i \leq p$ and $1 \leq j \leq t_i$, and let h be the number of occurrences of the clause $c_{d(i,j)}$ found in the sequence $c_{d(1,1)}, c_{d(1,2)}, \ldots, c_{d(1,t_1)}, c_{d(2,1)}, \ldots, c_{d(i,j)}$. Note that we must have $1 \leq h \leq 3$. Then we set $\pi(e(i,j)) = 3 \cdot [d(i,j) - 1] + h$.

We can now define the target instance $\langle G, [w, w']\rangle$ of our reduction. Let $[w, w'] = [w_1 w_2 \cdots w_p,\ c_1 c_2 \cdots c_n]$. Let also $G = (V_N, V_T, P, S)$, with $V_N = \{S\} \cup \{A_i \mid 1 \leq i \leq 3n\}$ and $V_T = C \cup \{a_i, b_i \mid 1 \leq i \leq p\}$. The productions below define set P:

(i) for every $1 \leq i \leq p$:

(a) for $1 \leq h \leq s_i$:

$$[A_{e(h,i)} \rightarrow x(i,h),\ A_{e(h,i)} \rightarrow c_{e(i,h)}],$$
$$[A_{e(h,i)} \rightarrow x(i,h),\ A_{e(h,i)} \rightarrow \varepsilon],$$
$$[A_{e(h,i)} \rightarrow \overline{x}(i,h),\ A_{e(h,i)} \rightarrow \varepsilon];$$

(b) for $s_i + 1 \leq h \leq t_i$:

$$[A_{e(h,i)} \rightarrow x(i,h),\ A_{e(h,i)} \rightarrow \varepsilon],$$
$$[A_{e(h,i)} \rightarrow \overline{x}(i,h),\ A_{e(h,i)} \rightarrow c_{e(i,h)}],$$
$$[A_{e(h,i)} \rightarrow \overline{x}(i,h),\ A_{e(h,i)} \rightarrow \varepsilon];$$

(ii)
$$[S \rightarrow A_{e(1,1)}^{(e(1,1))} A_{e(1,2)}^{(e(1,2))} \cdots A_{e(1,t_1)}^{(e(1,t_1))} A_{e(2,1)}^{(e(2,1))} \cdots A_{e(p,t_p)}^{(e(p,t_p))},$$
$$S \rightarrow A_{\pi(e(1,1))}^{(\pi(e(1,1)))} A_{\pi(e(1,2))}^{(\pi(e(1,2)))} \cdots A_{\pi(e(1,t_1))}^{(\pi(e(1,t_1)))} A_{\pi(e(2,1))}^{(\pi(e(2,1)))} \cdots A_{\pi(e(p,t_p))}^{(\pi(e(p,t_p)))}].$$

It is easy to see that $|G|$, $|w|$ and $|w'|$ are polynomially related to $|U|$ and $|C|$. From a derivation of $[w, w'] \in T(G)$, we can exhibit a truth assignment that satisfies C simply by reading off the derivation of the left string $w_1 w_2 \cdots w_p$. Conversely, starting from a truth assignment that satisfies C we can prove $w \in L(G)$ by means of (finite) induction on $|U|$: this part requires a careful inspection of all items in the definition of G. ■

From Theorem 1 we may conclude that algorithms for the membership problem for SCFGs are very unlikely to run in polynomial time. In the literature, several algorithms for this problem have been proposed using tabular methods (chart parsing). In the worst case, all these algorithms run in time $\Theta(|G| \cdot n^{k(G)})$, with G an SCFG and n the length of the input string pair. We know that, unless P = NP, $k(G)$ cannot be a constant. We now prove a lower bound on $k(G)$, providing thereby an exponential time lower bound result for our problem under the assumption of the tabular paradigm.

Tabular methods for the membership problem are based on the following representation. Given a synchronous production

$$\begin{aligned} s: \quad & [A_1 \to B_{11}^{(1)} \cdots B_{1r}^{(r)}, \\ & A_2 \to B_{21}^{(\pi(1))} \cdots B_{2r}^{(\pi(r))}], \end{aligned} \tag{1}$$

the already recognized constituent pairs $B_{1i}, B_{2\pi(i)}$ are gather together in several steps, keeping a record of the spanned substrings of the input. To provide a concrete example, if we gather all the B_{1i}'s on the left dimension from left to right, the partial analysis we obtain after the first step can be represented as a **state** $\langle s(1), (i_{11}, j_{11}), (i_{21}, j_{21})\rangle$, meaning that B_{11} and $B_{2\pi(1)}$ span substrings $w_1[i_{11}, j_{11}]$ and $w_2[i_{21}, j_{21}]$, respectively.[1] At the second step we have a state $\langle s(2), (i_{11}, j_{12}), (i_{21}, j_{21}), (i_{22}, j_{22})\rangle$, meaning that $B_{11}B_{12}$ together span $w_1[i_{11}, j_{12}]$, $B_{2\pi(1)}$ spans $w_2[i_{21}, j_{21}]$ and $B_{2\pi(2)}$ spans $w_2[i_{22}, j_{22}]$. We can see that, for some worst case permutations, the left-to-right strategy demands for increasingly more pairs of indices, so that the exponent in the time complexity linearly grows with r.

How much better can we do, if we exploit some strategy other than the left-to-right above? More precisely, we ask how many unconnected spannings a state may require for some worst case permutation π, under the choice of the best possible parsing strategy for π itself.

Theorem 2 *In the worst case, standard tabular methods for the SCFG membership problem require an amount of time $\Omega(|G|\, n^{c\cdot\sqrt{r}})$, with r the length of the longest production in G and c a constant.*

Proof. For any $r \geq 8$ we let $q = \lfloor\sqrt{r/2}\rfloor \geq \lfloor\sqrt{8/2}\rfloor = 2$, and define a permutation π_r on $\{1, \ldots, r\}$. We view the domain of π_r as composed of $2q$ **blocks** with q adjacent integers each, possibly followed by $r - 2q^2$ additional "padding" integers, and its codomain as composed of q blocks with $2q$ adjacent integers each, again possibly followed by $r - 2q^2$ "padding" integers. Permutation π_r transposes all blocks by sending the j-th element of the i-th block in the domain into the i-th element of the j-th block in the codomain, while mapping each padding integer identically into itself. Formally, for all positive integers $i \leq 2q$ and $j \leq q$, $\pi_r(q \cdot (i-1) + j) = 2q \cdot (j-1) + i$, and for all integers i with $2q^2 < i \leq r$, $\pi_r(i) = i$.

We count below how many spans are instantiated by a state that has gathered p constituent pairs, $1 \leq p \leq r$, in parsing production (1) under any possible strategy. When a constituent pair $B_{1i}, B_{2\pi_r(i)}$ is gathered, we say integer i in the domain of π_r and integer $\pi_r(i)$ in the codomain have been pebbled. In this way each span (i, j) in a state corresponds to some run $i, i+1, \ldots j$ of pebbled integers, with either $i = 1$ or $i - 1$ unpebbled, and with either $j = r$ or $j + 1$ unpebbled. We call each such run a **segment**, and show that every parsing strategy demands at least $q = \lfloor\sqrt{r/2}\rfloor$ segments either in the domain or in the codomain of π_r.

We say that a block in the domain of π_r is empty, full, or mixed if, respectively, none, all, or some but not all of its elements have been pebbled. Assume that, for a given parsing strategy, the last block that becomes mixed does so when we place the i-th pebble, and the first block that becomes full does so when we place the j-th pebble. Obviously $i \neq j$: the first pebble placed in a previously empty block can not make it full since every block contains at least 2 elements.

If $i < j$, after placing the i-th pebble and before placing the j-th pebble every block in the domain of π_r is mixed. Each of these $2q$ blocks then contains at least one pebbled element which is adjacent to an unpebbled one and must therefore be either the first or the last element of a segment. The domain of π_r then contains at least $2q/2 = q$ segments.

If $j < i$, after placing the j-th pebble and before placing the i-th pebble at least one block in the domain of π_r (e.g., the h-th block) is full, and at least one (e.g., the k-th) is empty. Then, in each of the q blocks in the codomain of π_r, the h-th element is pebbled while the k-th is not. Therefore the h-th elements of any two consecutive blocks in the codomain of π_r must belong to two distinct segments, since at least one intermediate element is not

[1]For a string $w = a_1 \cdots a_n$, we write $w[i, j]$ to denote the substring $a_{i+1} \cdots a_j$.

pebbled. The codomain of π_r then contains at least q segments. ■

4 The translation problem

In this section we consider some formulations of the translation problem for PSCFG that have been proposed in the literature. The most general definition of the **translation problem** for PSCFG is this: for an input PSCFG $G_p = (G, p_G)$ and an input string w, produce a representation of all possible parse trees, along with their probabilities, that are assigned by G to a string in the set $T(G, w)$ under some translation of w.

Variant of this definition can be found where the input is a single parse tree for w (Yamada and Knight, 2001), or where the output is a single parse tree, chosen according to some specific criteria (Wu and Wong, 1998). To formally study these problems, in what follows we focus on single parse trees associated with derivations in G_p. For a derivation σ of the form $[S^{(1)}, S^{(1)}] \Rightarrow^{\sigma}_{G} [w_1, w_2]$, we write $t_{\sigma,l}$ and $t_{\sigma,r}$ to denote the left and the right parse trees, respectively, associated with σ. The probability that $t_{\sigma,r}$ is obtained as a translation of $t_{\sigma,l}$ through G_p is thus $p_G([t_{\sigma,l},\ t_{\sigma,r}]) = p_G(\sigma)$. Let t be some parse tree; we write $y(t)$ to denote the string in the yield of t. For a string $w \in V_T^*$ and a parse tree t, we also consider the probability that t is obtained from w through G_p, defined as:

$$p_G([w,\ t]) \ = \sum_{y(t')=w} p_G([t',\ t]). \qquad (2)$$

We can now precisely define the variants of the translation problem we are interested in. Given as input a PSCFG $G_p = (G, p_G)$ and two strings $w_1, w_2 \in V_T^*$, output the pair of parse trees

$$\operatorname*{argmax}_{\substack{y(t_1)=w_1,\\ y(t_2)=w_2}} p_G([t_1,\ t_2]). \qquad (3)$$

If the synchronous productions in the underlying SCFG G have length bounded by some constant, then the above problem can be solved in polynomial time using extensions of the Viterbi search strategy to parse forests. This has been shown for instance in (Wu and Wong, 1998; Yamada and Knight, 2001; Melamed, 2004).

A second interesting problem is defined as follows. Given as input a PSCFG $G_p = (G, p_G)$ and a string $w \in V_T^*$, output the parse tree

$$\operatorname*{argmax}_{t} p_G([w,\ t]). \qquad (4)$$

Even in case we impose some constant bound on the length of the synchronous productions in G, the above problem is NP-hard, as we show in what follows.

We assume the reader is familiar with the definition of probabilistic context-free grammar (PCFG) and with the associated notion of derivation probability (Wetherell, 1980). We denote a PCFG as a pair (G, p_G), with $G = (V_N, V_T, P, S)$ the underlying context-free grammar and p_G the associated function providing the probability distributions for the productions in P, conditioned on their left-hand side. A probabilistic regular grammar (PRG) is a PCFG with underlying productions of the form $A \rightarrow aB$ or $A \rightarrow \varepsilon$, with A, B nonterminal symbols and a a terminal symbol.

We consider below a decision problem associated with PRG, called the **consensus** problem, defined as follows: Given as input a PRG (G, p_G) and a rational number $d \in [0, 1]$, decide whether there exists a string w in the language generated by G such that $p_G(w) \geq d$. It has been shown in (Casacuberta and de la Higuera, 2000) that, for a PRG G whose productions have all probabilities expressed by rational numbers, the above problem is NP-complete. (Essentially the same result is also reported in (Lyngso and Pedersen, 2002), stated in terms of hidden Markov models.) We reduce the consensus problem for PRG to a decision version of the problem in (4), called the **best translated derivation** problem and defined as follows. Given as input a PCFG $G_p = (G, p_G)$, a string $w \in V_T^*$ and a rational number $d \in [0, 1]$, decide whether $\max_t\ p_G([w,\ t]) \geq d$.

Theorem 3 *The best translated derivation problem for the class PSCFG is NP-hard.*

Proof. We provide a reduction from the consensus problem for the class PRG with rational production probabilities. The main idea is described in what follows. Given the input PRG G_p, we construct a target PSCFG G'_p that translates string \$ into \$, with \$ a special symbol. Given as input the string \$, G'_p simulates all possible derivations of G_p through its own derivations. This is done by encoding the nonterminals appearing in a derivation ρ of G_p within the left component of some derivation σ of G'_p, and by encoding the terminal string generated by ρ within the right component of σ. The probability of ρ is also preserved by σ.

Let $G_p = (G, p_G)$, d be an instance of the consensus problem as above, with $G = (V_N, V_T, P, S)$. We specify a PSCFG $G'_p = (G', p_{G'})$ with $G' = (V'_N, \{\$\}, P', S)$ and $V'_N = V_N \cup V_T$. Set P' is constructed as follows:

(i) for every $(S \rightarrow aA) \in P$, $s : [S \rightarrow A^{(1)},\ S \rightarrow a^{(1)}]$ is added to P', with $p_{G'}(s) = p_G(S \rightarrow aA)$;

(ii) for every $(S \rightarrow \varepsilon) \in P$, $s : [S \rightarrow \$,\ S \rightarrow \$]$ is added to P', with $p_{G'}(s) = p_G(S \rightarrow \varepsilon)$;

(iii) for every $a \in V_T$ and $(A \rightarrow bB) \in P$, $s : [A \rightarrow B^{(1)},\ a \rightarrow b^{(1)}]$ is added to P', with $p_{G'}(s) = p_G(A \rightarrow bB)$

(iv) for every $a \in V_T$ and $(A \rightarrow \varepsilon) \in P$, $s : [A \rightarrow \$,\ a \rightarrow \$]$ is added to P', with $p_{G'}(s) = p_G(A \rightarrow \varepsilon)$.

Note that the construction of G'_p can be carried out in quadratic time in the size of G_p. It is not difficult to see that there exists a derivation of the form $S \Rightarrow_G a_1 A_1 \Rightarrow_G a_1 a_2 A_2 \cdots \Rightarrow_G a_1 a_2 \cdots a_n A_n$ if and only if there exist a derivation in G' associated with unary trees t_1 and t_2, such that string $S A_1 A_2 \cdots A_n$ is read from the spine of t_1 and string $S a_1 a_2 \cdots a_n$ is read from the spine of t_2. Furthermore, the two derivations are composed of 'corresponding' productions with the same probabilities. We conclude that there exists a string w in $L(G)$ with $p_G(w) > d$ if and only if there exists a unary tree t with string $Sw\$$ read from the spine such that $p_{G'}([\$, t]) > d$. ■

We discuss below an interesting consequence of Theorem 3. The SDTS formalism discussed in Section 1 has been extended to the probabilistic case in (Maryanski and Thomason, 1979), called stochastic SDTS (SSDTS). As a corollary to the proof of Theorem 3, we obtain that one can define, through some PSCFG G_p and some fixed string w, a probability distribution $p_G([w, t])$ on parse trees that cannot be obtained through any SSDTS. Without providing the details of the definition of SSDTS, we give here only an outline of the proof. We also assume that the reader is familiar with probabilistic finite automata and with their distributional equivalence with PRG.

Consider the PSCFG $G'_p = (G', p_{G'})$ defined in the proof of Theorem 3, and assume there exists some SSDTS $G''_p = (G'', p_{G''})$ such that, for every tree t, we have $p_{G''}([\$, t]) = p_{G'}([\$, t])$. Since in a derivation of an SDTS the generated trees are always isomorphic, up to some reordering of sibling nodes, we obtain that the productions of G'' must have the form $[S \rightarrow a^{(1)},\ S \rightarrow a^{(1)}]$, $[a \rightarrow b^{(1)},\ a \rightarrow b^{(1)}]$ and $[a \rightarrow \$,\ a \rightarrow \$]$. From these productions we can construct a probabilistic deterministic finite automaton generating the same language as the PRG G_p, and with the same distribution. But this is impossible since there are string distributions defined by some PRG that cannot be obtained through probabilistic deterministic finite automata; see for instance (Vidal et al., 2005).

We conclude by remarking that in (Casacuberta and de la Higuera, 2000) it is shown that finding the best output string for a given input string is NP-hard for stochastic SDTS with a single nonterminal in each production's right-hand side. Our result in Theorem 3, stated for PSCFG, is stronger, since it investigates individual parse trees rather than strings.

5 Concluding remarks

The presented results are based on worst case analysis: further experimental evaluation needs to be carried out on multilingual corpora in order to asses the practical impact of these findings.

Acknowledgment

We are indebted to Dan Melamed and Mark-Jan Nederhof for technical discussion on topics related to this paper. Dan Melamed also suggested to us the problem investigated by Theorem 2. The first author is partially supported by MIUR under project PRIN No. 2003091149_005.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1):45–60, March.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, Robert L. Mercer, and Paul Roossin. 1988. A statistical approach to language translation. In *Proceedings of the International Conference on Computational Linguistics (COLING) 1988*, pages 71–76, Budapest, Hungary, August.

F. Casacuberta and C. de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications; 5th International Colloquium, ICGI 2000*, pages 15–24. Springer.

D. Chiang. 2004. *Evaluating Grammar Formalisms for Applications to Natural Language Processing and Byological Sequence Analysis*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43^{rd} ACL*, pages 263–270.

M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability*. Freeman and Co., New York, NY.

Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan, July.

Y. Kaji et al. 1994. The computational complexity of the universal recognition problem for parallel multiple context-free grammars. *Computational Intelligence*, 10(4):440–452.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics, Squibs and Discussion*, 25(4).

S. Kumar and W. Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of HLT-NAACL*.

R. B. Lyngso and C. N. S. Pedersen. 2002. The consensus string problem and the complexity of comparing hidden markov models. *Journal of Computing and System Science*, 65:545–569.

Fred J. Maryanski and Michael G. Thomason. 1979. Properties of stochastic syntax-directed translation schemata. *International Journal of Computer and Information Sciences*, 8(2):89–110.

I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 158–165, Edmonton, Canada.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, July.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the 4nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 20–28, College Park, Maryland.

G. Satta. 1992. Recognition of linear context-free rewriting systems. In *Proc. of the 30^{th} ACL*, Newark, Delaware.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. 2005. Probabilistic finite-state machines – Part I. *IEEE Trans. on Pattern analysis and Machine Intelligence*. To appear.

C. S. Wetherell. 1980. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Montreal, Canada, July.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, September.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 531–538, Toulouse, July.

Incremental LTAG Parsing

Libin Shen and **Aravind K. Joshi**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{libin,joshi}@linc.cis.upenn.edu

Abstract

We present a very efficient statistical incremental parser for LTAG-spinal, a variant of LTAG. The parser supports the full *adjoining* operation, dynamic predicate coordination, and non-projective dependencies, with a formalism of provably stronger generative capacity as compared to CFG. Using gold standard POS tags as input, on section 23 of the PTB, the parser achieves an f-score of 89.3% for syntactic dependency defined on LTAG derivation trees, which are deeper than the dependencies extracted from PTB alone with head rules (for example, in Magerman's style).

1 Introduction

Lexicalized Tree Adjoining Grammar (LTAG) is a formalism motived by both linguistic and computational perspectives (for a relatively recent review, see (Joshi and Schabes, 1997)). Because of the introduction of the *adjoining* operation, the TAG formalism is provably stronger than Context Free Grammar (CFG) both in the weak and the strong generative power. The TAG formalism provides linguistically attractive analysis of natural language (Frank, 2002). Recent psycholinguistic experiments (Sturt and Lombardo, 2005) demonstrate that the *adjoining* operation of LTAG is required for eager incremental processing.

Vijay-Shanker and Joshi (1985) introduced the first TAG parser in a CYK-like algorithm. Because of the adjoining operation, the time complexity of LTAG parsing is as large as $O(n^6)$, compared with $O(n^3)$ of CFG parsing, where n is the length of the sentence to be parsed. Many LTAG parsers were proposed, such as the head-driven Earley style parser (Lavelli and Satta, 1991) and the head-corner parser (van Noord, 1994). The high time complexity prevents LTAG parsing from real-time applications.

In this paper, we work on LTAG-spinal (Shen and Joshi, 2005), an interesting subset of LTAG, which preserves almost all of the strong generative power of LTAG, and it is both weakly and strongly more powerful than CFG [1]. We will present a statistical incremental parsing for LTAG-spinal. As far as we know, this parser is the first comprehensive attempt of efficient statistical parsing with a formal grammar with provably stronger generative power than CFG, supporting the full *adjoining* operation, dynamic predicate coordination, as well as non-projective dependencies [2].

2 LTAG-spinal and the Treebank

We first briefly describe the LTAG-spinal formalism and the LTAG-spinal treebank to be used in this paper. More details are reported in (Shen and Joshi, 2005).

In LTAG-spinal, we have two different kinds of elementary trees, *initial* trees and *auxiliary* trees, which are the same as in LTAG. However, as the name implies, an initial tree in LTAG-spinal only contains the *spine* from the root to the anchor, and an auxiliary tree only contains the spine and the foot node directly connected to a node on the spine.

Three types of operations are used to connect the elementary trees into a derivation tree, which are *attachment*, *adjunction* and *conjunction*. We show LTAG-spinal elementary trees and operations with an example in Figure 1.

In Figure 1, each arc is associated with a character which represents the type of operation. We use **T** for a*t*tach, **A** for *a*djoin, and **C** for *c*onjoin.

[1]Further formal results are described in (Shen and Joshi, 2005). There is also some relationship of LTAG-spinal to the spinal form context-free tree grammar, as in (Fujiyoshi and Kasai, 2000)

[2]In (Riezler et al., 2002), the MaxEnt model was used to rerank the K-best parses generated by a rule-based LFG parser.

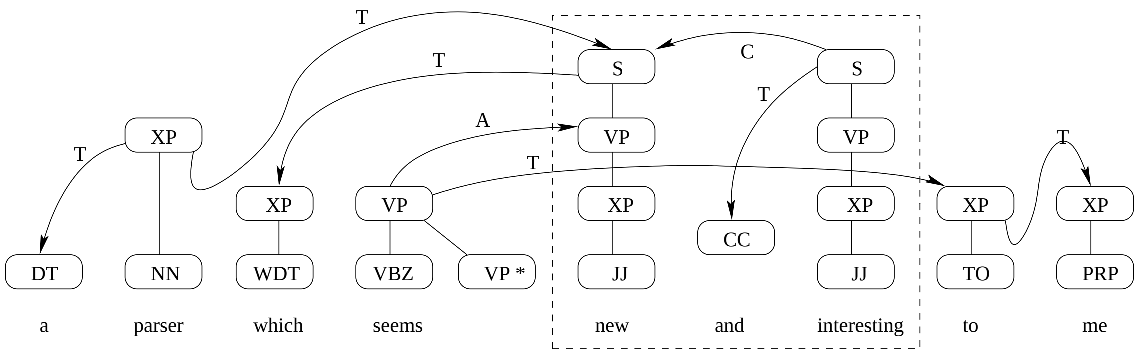


Figure 1: An example in LTAG-spinal. **A**=adjoin, **T**=attach, **C**=conjoin.

Attachment in LTAG-spinal is similar to *sister adjunction* (Chiang, 2000) in Tree Insertion Grammar (TIG) (Schabes and Waters, 1995). It represents a combination of *substitution* and *sister adjunction*. The *attachment* operation is designed to encode the ambiguity of an *argument* and an *adjunct*.

Adjunction inserts part of the spine and the foot node of an auxiliary tree into to the spine of another tree. The *adjunction* operation can effectively do wrapping, which distinguishes itself from *sister adjunction*. It is not difficult to see that adjunction only happens on the spine of a tree. This property will be exploited in the incremental parser.

Conjunction is similar to what was originally proposed in (Sarkar and Joshi, 1996). However, in LTAG-spinal, the conjunction operation is much easier to handle, since we only conjoin spinal elementary trees and we do not need to enumerate *contraction sets* for conjunction. In our formalization, *conjunction* can be treated as a special *adjunction*, however, this is beyond the scope of this paper.

We use the LTAG-spinal treebank described in (Shen and Joshi, 2005), which was extracted from the Penn Treebank (PTB) (Marcus et al., 1994) with Propbank (Palmer et al., 2005) annotations.

2.1 Relation to Traditional LTAG

LTAG-spinal preserves most of the strong generative power of LTAG. It can be shown that LTAG-spinal with adjoining restrictions (Joshi and Schabes, 1997) has stronger generative capacity as compared to CFG. For example, there exists an LTAG-spinal grammar that generates $\{a^n b^n c d^n e^n \mid n > 0\}$, which is not a context-free language.

A spinal elementary tree is smaller than a traditional LTAG elementary tree which contains all the substitution nodes of the arguments. In the LTAG-spinal formalism, both arguments and adjuncts are expected to be directly attached or adjoined onto a spine. In this sense, LTAG-spinal roughly satisfies *the fundamental TAG hypothesis: Every syntactic dependency is expressed locally within a single elementary tree* (Frank, 2002). The only difference is that, in LTAG-spinal, syntactic dependencies are represented via direct or local connections.

To better understand the meaning of this difference, we relate it to Frank's (2002) model for how the LTAG elementary trees are constructed. In Frank's model, all the elementary trees are built via *Marge* and *Move* operations, starting with a *local lexical array*. The resulting LTAG elementary trees are then combined with adjunction and substitution to build a derivation tree.

Thus, in a sense, the LTAG-spinal grammar opens a door to a *parallel* mechanism of building the elementary trees and the derivation tree. The spinal templates in LTAG-spinal only contain the path of projection from the anchor to the top node. A spinal template plus the root nodes of the subtrees attached to this template can be viewed as a traditional LTAG elementary tree. More specifically, it encodes a set of possible elementary trees if we distinguish substitution from sister adjunction. Thus, the LTAG-spinal parsing model to be proposed in Section 3 can be viewed as a parser at the meta-grammar (Candito, 1998; Kinyon and Prolo, 2002) level for traditional LTAG. Derivation tree construction and full-size elementary tree filtering are processed in parallel. Researches in statistical CFG parsing (Ratnaparkhi, 1997; Collins, 1999) and psycholinguistics

(Shieber and Johnson, 1993) showed that this strategy is desirable for NLP.

Furthermore, the way that we split a traditional LTAG elementary tree along the spine is similar to the method with which Evans and Weir (1997) compiled the XTAG English Grammar into finite state automata. In their work, this method was designed to employ shared structure in a rule-based parser. But here we extend this technique to statistical LTAG parsing.

2.2 Relation to Propbank

In building the LTAG-spinal Treebank, the Propbank information is used in the treebank extraction. As reported in (Shen and Joshi, 2005), tree transformation on PTB are employed to make it more compatible with the Propbank annotations. It was shown that 8 simple patterns of the path from a predicate to an argument account for 95.5% of the total pred-arg pairs. Thus, our high-quality parsing output will be very useful for semantic role labeling.

Arguments in Propbank are not *obligatory* complements. Therefore, we cannot treat the Propbank arguments as the *arguments* in LTAG. The ambiguity of argument and adjunct is reflected in the similarity of substitution and sister adjunction. This is one of the reasons that we do not distinguish substitution and sister adjunction in LTAG-spinal.

3 Incremental Parsing

We are especially interested in incremental parsing for the following two reasons. Firstly, the left to right strategy used in incremental parsing gives rise to a drastic boost in speed. Furthermore, there is also a strong connection between incremental parsing and psycholinguistics, and this connection is also observed in the LTAG formalism (Ferreira, 2000; Sturt and Lombardo, 2005).

In recent years, there have been many interesting works on incremental or semi-incremental parsing. By semi-incremental we mean the parsers that allow several rounds of left to right scans instead of one. Both left-corner strategy (Ratnaparkhi, 1997; Roark, 2001; Prolo, 2003; Henderson, 2003; Collins and Roark, 2004) and head-corner strategy (Henderson, 2000; Yamada and Matsumoto, 2003) were employed in incremental parsing. The head-corner approach is more natural to the LTAG formalism (Evans and Weir, 1997). In our approach, we use a stack of derivation treelets to represent the partial parsing result. Furthermore, the LTAG formalism allows us to handle non-projectivity dependencies, which cannot be generated by a CFG or a Dependency parser.

In fact, the idea of incremental parsing with LTAG is closely related to the work on Supertagging (Joshi and Srinivas, 1994). A supertager first assigns the correct LTAG elementary tree to each word. Then a Lightweight Dependency Analyzer (LDA) (Srinivas, 1997) composes the whole derivation tree with these elementary trees. We use incremental parsing to incorporate supertager and LDA dynamically.

The model of incremental LTAG parsing is also similar to Structured Language Modeling (SLM) in (Chelba and Jelinek, 2000). In SLM, the left context of history is represented with a stack of binary trees. At each step, one computes the likelihood of the current word, its tag and the operations over the new context trees.

3.1 Treatment of Coordination

Predicate coordination appears in about 1/6 of the sentences in PTB, therefore proper treatment of coordination, especially predicate coordination, is important to parsing of PTB.

Some recent results in psycholinguistic experiments (Sturt and Lombardo, 2005) showed a high degree of *eagerness* in building coordination structures which is absent in a bottom-up approach; A bottom-up parser waits for the second conjunct to be completed before combining the two conjuncts as for example in VP coordination, and then combine the coordinated VP with the subject of the left conjunct. Psycholinguistic results suggest that the right conjunct has to have access to the subject NP of the left conjunct. This can be achieved by first building the entire S on the left and then *adjoining* the right VP conjunct to the VP node of the left conjunct (Sturt and Lombardo, 2005).

We follow the strategy suggested by the psycholinguistic experiments, treating conjoining as a special adjoining operation.

3.2 The Parsing Algorithm

There are four different types of operations in our parser. Three of them are described in Section 2. The fourth operation is **generation**, which is used to generate a possible spine for a given word according to the context and the lexicon.

Our left to right parsing algorithm is a variant of the shift-reduce algorithm with beam-search. We use a stack of disconnected derivation treelets to represent the left context. When the parser reads a word, it first *generates* a list of possible spinal elementary trees for this word, For each elementary tree, we first push it into the stack. Then we recursively pop the top two treelets from the stack and push the combined tree into the stack until we choose not to combine the top two treelets with one of the three combination operations (we can also choose not to pop anything at the beginning). Then we shift to the next word. This model is called the **Flex Model** in this paper.

A potential problem with the Flex Model is that a single *LTAG derivation* tree can be generated by several *shift-reduce derivation* steps, which only differ in the order of operations. For example, we have three trees A, B and C. In LTAG derivation, A adjoins to B, and B adjoins to C. Then we have two different shift-reduce derivations, which are $(A \rightarrow (B \rightarrow C))$ and $((A \rightarrow B) \rightarrow C)$.

Now we introduce the **Eager Model**, an eager evaluation strategy. Any two elementary trees which are directly connected in the LTAG derivation tree are combined immediately when they can be combined in some context. Furthermore, they cannot be combined afterwards, if they miss the first chance. In the previous example, the parser will generate $((A \rightarrow B) \rightarrow C)$, while $(A \rightarrow (B \rightarrow C))$ is ruled out. Then for each LTAG derivation tree, there exists a unique left-to-right derivation.

The Eager Model is motived by the treatment of coordination in (Sturt and Lombardo, 2005), as we discussed in the previous section. For example, we have the following two sentences.

1. Quimby knows Tom likes Philly steak.
2. Quimby knows Tom likes Philly steak and Jerry likes pizza.

Suppose we are parsing these two sentences, and for each case the current word is *likes*, the fourth word. Now we have just the same local contexts for both cases. According to the Eager Model, the parser takes the same action according to the context, which is to combine the *knows* tree and the *likes* tree. For sentence 2, the second *likes* tree will be conjoined with the first *likes* tree later. This is compatible with the psycholinguistic preference.

In the following section, we will explain the parsing mechanism for the Eager Model with an example. The Flex Model is similar except that the order of operations is flexible to some extent.

3.3 An Example

Figure 2 shows the left to right parsing of the phrase *a parser which seems new and interesting to me* with the Eager Model.

In Figure 2, each arc is associated with a number and a character. The number represents the order of operation, and the character stands for the type of operation as in Figure 1. Furthermore we use **G** to represent Generate.

In step 1 and 2, two disconnected spines are generated for *a* and *parser*. The spine for *a* is attached to the spine for *parser* on the *NP* node in step 3.

In step 6, the spine for *new*, the first conjunct of the predicate coordination, is generated. Then the auxiliary tree for *seems* is adjoined to the spine for *new* at the node *VP*. the latter is further combined with *which*, and is attached to the tree for *parser*.

In step 13, the conjoin operation is used to combine the treelet anchored on *new* and the treelet anchored on *interesting*. Alignments between the two spines are built, through which argument sharing is implemented in an implicit and underspecified way.

In step 15, for the spine for *to*, the visible nodes of the conjoined treelet include nodes on some auxiliary trees adjoined on the left of the spines, like the root *VP* node for *seems*. In this way, a non-projective structure is generated, which is just the same as the wrapping adjoining in LTAG.

3.4 Machine Learning Algorithm

Many machine learning algorithms have been successfully applied to parsing, incremental parsing, or shallow parsing (Ratnaparkhi, 1997; Punyakanok and Roth, 2001; Lafferty et al., 2001; Taskar et al., 2003), which can be applied to our incremental parsing algorithm.

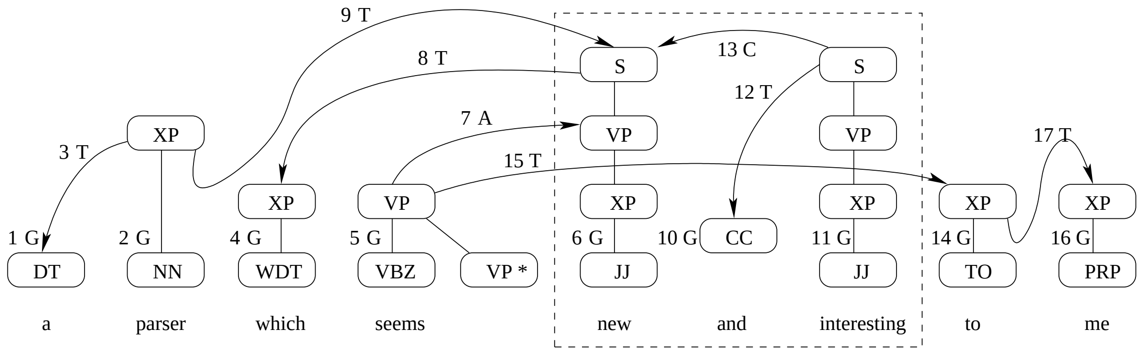


Figure 2: Incremental parsing with Eager Model. **A**=adjoin, **T**=attach, **C**=conjoin, **G**=generate

In this paper, we use the perceptron-like algorithm proposed in (Collins, 2002) which does not suffer from the label bias problem, and is fast in training. We also employ the voted perceptron algorithm (Freund and Schapire, 1999) and the *early update* technique as in (Collins and Roark, 2004).

3.5 Features

Features are defined in the format of (*operation, main spine, child spine, spine node, context*), where the *spine node* is the node on the *main spine* onto which the *child spine* is *attached* or *adjoined*. For *generate*, child spine and spine node are undefined, and for *conjoin* spine node is undefined. *context* describes the constituent label or lexical item associated with a certain node. The context of an operation includes the top two treelets involved in the operation as well as the two closest words on both sides of the current word.

- Context for *generate* :
  - The (-2, 2) window in the flat sentence.
  - The *visible* [3] spines on the topmost treelet.
- Context for *attach* and *adjoin* :
  - The (0, 2) window in the flat sentence.
  - The most recent spine previously attached or adjoined to the same location on the main spine.
  - The leftmost child spine attached to the child spines.
  - The spines that are *visible* before the operation and become *invisible* after the operation.
- Context for *conjoin* :
  - The (0, 2) window in the flat sentence.
  - The leftmost child spine attached to the main spine, which is the first adjunct.
  - The two leftmost children spines attached to the child spine, which is the current adjunct.

[3]The details are presented in (Shen, 2005).

We have about 1.4M features extracted from the gold-standard parses, and about 600K features dynamically extracted from the generated parses in 10 rounds of training with the Eager Model.

4 Experiments and Analysis

We use the LTAG-spinal treebank reported in (Shen and Joshi, 2005). The LTAG-spinal parse for the 39434 sentences extracted from WSJ section 2-21 are used as the training data. Section 24 is used as the development data. Section 23 are used for test[4].

We use syntactic dependency for evaluation. It is worth mentioning that, for predicate coordination, we define the dependency on the parent of the coordination structure and each of the conjunct predicate. For example, in Figure 1, we have dependency relation on (parser, *new*) and (parser, *interesting*). Compared with other dependency parsers on PTB, the dependency defined on LTAG-spinal reveals deeper relations because of the treatment of traditional adjoining and predicate coordination described above.

In the community of parsing, *labeled recall* and *labeled precision* on phrase structures are often used for evaluation. However, in our experiments we cannot evaluate our parser with respect to the phrase structures in PTB. As shown in (Shen and Joshi, 2005), various irrecoverable tree transformations

[4]The LTAG-spinal treebank contains 2401 out of 2416 sentences in section 23.

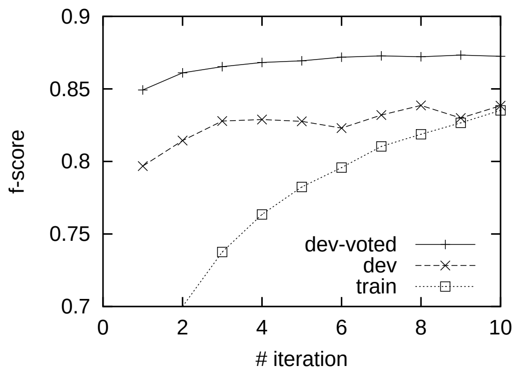


Figure 3: f-score of syntactic dependency on the *development* data with the Eager Model

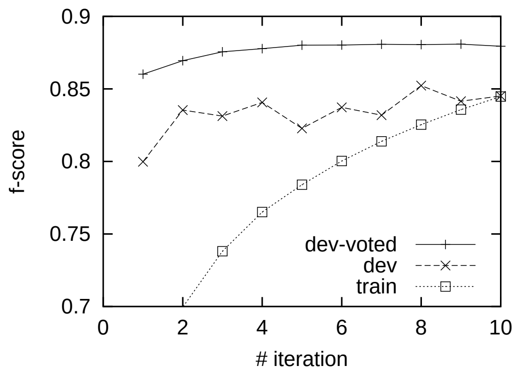


Figure 4: f-score of syntactic dependency on the *development* data with the Flex Model

were used to extract the LTAG-spinal treebank according the Propbank annotation on PTB. Therefore, we use syntactic dependency for evaluation.

4.1 Eager vs. Flex

We first train our incremental parser with Eager Model and Flex Model respectively. In the training, beam width is set to 10. Lexical features are limited to words appearing for at least 5 times in the training data. Figure 3 and Figure 4 show the learning curves on the training and the development data. The X axis represents the number of iterations of training, and the Y axis represents the f-score of dependency with respected to the LTAG derivation tree.

Since *early update* is used, the f-score on the training data is very low at the beginning. In both cases, the voted weights provide an f-score which is more than 3% higher. The voted results converge faster and are more stable. The result with Flex Model is 0.6% higher than the one with Eager Model, but the parsing time is much longer with Flex Model as we will show later.

We use the voted weights obtained after 10 rounds of iteration for the evaluation on the test data. We achieve an f-score of **88.7%** on dependency with the Eager Model, and **89.3%** with the Flex Model. The Flex Model achieves better performance because it allows the decision of operation to be delayed until there is enough context information.

4.2 K-Best Parsing

The next experiment is on K-best parsing. As a first attempt, we just use the same algorithm as in the previous section, except that we study the oracle parse, or the best parse, among the top 10 parses. The f-score on the oracle in top 10 in the development data is 88.5%, while the f-score of the top candidate is 87.3%, as shown in Table 1. However, we are not satisfied with the score on oracle, which is not good enough for post-processing, i.e. parse reranking.

Table 1: F-score of the oracle parse in the 10-best parses on the *development* data with the Eager Model

algorithm	f-score%
top (eager)	87.3
oracle (eager)	88.5
top (eager+combined parses)	87.4
oracle (eager+combined parses)	91.0

We notice that from a single partial derivation we can generate a large set of different partial derivations, just by combining the elementary tree of the next word. It is easy to see that these similar derivations may use up the search beam quickly, which is not good for parse search. Many of the new derivations share the same dependency structure. So we revised our learning procedure by combining derivations with the same dependency structure before each shift operation. We repeated the K-best parsing experiments by using **Combined Parses** as described above, and achieved significant improvement on the oracle, as shown in Table 1.

Figure 5 shows the f-score of the oracle on K-best parsing using combined parses on the test data. For each K-best oracle test, we set the beam width to K

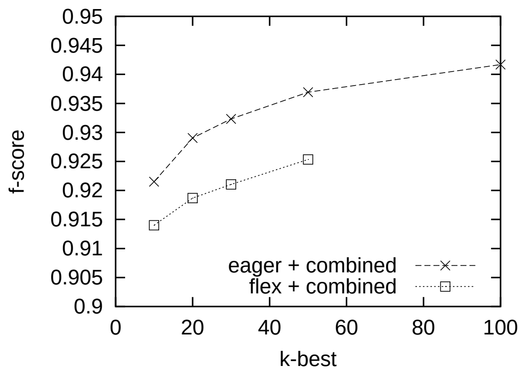


Figure 5: f-score of the oracle on the *test* data

Table 2: Speed of parsing on the *test* data set. Here cp? = whether the method of Combined Parses is used; sen/sec = sentence per second; top = top candidate given by the parser; oracle = oracle of the K-best parses where K equals the width of the beam.

model	cp?	beam	sen/sec	f-score%
single best				top
flex	no	10	0.37	89.3
eager	no	10	0.79	88.7
K-best				oracle
eager	yes	10	0.62	92.2
eager	yes	20	0.31	92.9
eager	yes	30	0.22	93.2
eager	yes	50	0.13	93.7
eager	yes	100	0.07	94.2

in parsing. The f-score of oracle in 100-best parsing is **94.2%** with the Eager Model + Combined Parses.

4.3 Speed of Parsing

Efficiency is important to the application of incremental parsing. This set of experiments is related to the speed of our parser on single best and K-best parsing with both the Eager Model and the Flex Model. All the experiments are performed on a Linux node with two 1.13GHz PIII CPUs and 2GB RAM. The parser is coded in Java.

Table 2 shows that the Eager Model is more than two times faster than the Flex Model, as we expected. The time spent on K-best parsing is proportional to the beam width.

5 Discussion and Future Work

The parser proposed in this paper is an incremental parser, so the accuracy on dependency is lower than that for chart parsers, for example like those reported in (Collins, 1999; Charniak, 2000).[5] However, it should be noted that the dependencies computed by our parser are *deeper* than those calculated by parsers working directly on PTB. This is due to the treatment of adjunction and coordination.

On the other hand, the LTAG-spinal treebank used in this paper shows a high degree of compatibility with the Propbank, as shown in (Shen and Joshi, 2005), so the LTAG derivations given by the parser are very useful for predicate-argument recognition. We plan to improve the parsing performance by reranking and extend our work to semantic parsing (Mooney, 2004).

Another interesting topic is whether this parser can be applied to languages which have various long-distance scrambling, as in German. It appears that by carefully modifying the definition of *visible* spines, we can represent scrambling structures, which at present can only be represented by Multi-Component TAG (Becker et al., 1991).

6 Conclusions

In this paper, we present an efficient incremental parser for LTAG-spinal, a variant of LTAG which is both linguistically and psycholinguistically motivated. As far as we know, the statistical incremental parser proposed in this paper is the first comprehensive attempt of efficient statistical parsing with a formal grammar with provably stronger generative power than CFG, supporting the *adjoining* operation, dynamic predicate coordination, as well as non-projective dependencies.

We have trained and tested our parser on the LTAG-spinal treebank, extracted from the Penn Treebank with Propbank annotation, Using gold standard POS tags as part of the input, the parser achieves an f-score of 89.3% for syntactic dependency on section 23 of PTB. Because of the treatment of adjunction and predicate coordination, These dependencies, which are defined on LTAG-spinal derivation trees, are deeper than the dependencies extracted from PTB alone with head rules.

[5]We plan to work on a chart parser for LTAG-spinal.

References

T. Becker, A. K. Joshi, and O. Rambow. 1991. Long distance scrambling and Tree Adjoining Grammars. In *EACL 1991*.

M. Candito. 1998. Building parallel ltag for french and italian. In *ACL-COLING 1998*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL 2000*.

C. Chelba and F. Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

D. Chiang. 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *ACL 2000*.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL 2004*.

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

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*.

R. Evans and D. Weir. 1997. Automaton-based parsing for lexicalized grammars. In *IWPT 1997*.

F. Ferreira. 2000. Syntax in language production: An approach using tree-adjoining grammars. In L. Wheeldon, editor, *Aspects of Language Production*. MIT Press.

R. Frank. 2002. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

A. Fujiyoshi and T. Kasai. 2000. Spinal-formed context-free tree grammars. *Theory Computing Systems*, 33(1).

J. Henderson. 2000. A neural network parser that handles sparse data. In *IWPT 2000*.

J. Henderson. 2003. Generative versus discriminative models for statistical left-corner parsing. In *IWPT 2003*.

A. K. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69 – 124. Springer.

A. K. Joshi and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *COLING 1994*.

A. Kinyon and C. Prolo. 2002. A classification of grammar development strategies. In *COLING 2002 Workshop: Grammar Engineering and Evaluation*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*.

A. Lavelli and G. Satta. 1991. Bidirectional parsing of lexicalized tree adjoining grammars. In *EACL 1991*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

R. Mooney. 2004. Learning semantic parsers: An important but under-studied problem. In *AAAI 2004 Spring Symposium on Language Learning: An Interdisciplinary Perspective*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

C. Prolo. 2003. *LR Parsing for Tree Adjoining Grammars and its Application to Corpus-based Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS 2001*.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *EMNLP 1997*.

S. Riezler, T. King, R. Kaplan, R. Crouch, J. Maxwell, and M. Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *ACL 2002*.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

A. Sarkar and A. K. Joshi. 1996. Coordination in tree adjoining grammars. In *COLING 1996*.

Y. Schabes and R. C. Waters. 1995. A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4).

L. Shen and A. K. Joshi. 2005. Building an LTAG treebank. Technical Report MS-CIS-05-15, CIS Dept., UPenn.

L. Shen. 2005. Statistical Natural Language Processing with Lexicalized Tree Adjoining Grammar. Ph.D. proposal, University of Pennsylvania.

S. Shieber and M. Johnson. 1993. Variations on incremental interpretation. *Journal of Psycholinguistic Research*, 22(2):287–318.

B. Srinivas. 1997. Performance evaluation of supertagging for partial parsing. In *IWPT 1997*.

P. Sturt and V. Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science, to appear*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *NIPS 2003*.

G. van Noord. 1994. Head corner parsing for TAG. *Computational Intelligence*, 10(4).

K. Vijay-Shanker and A. K. Joshi. 1985. Some computational properties of tree adjoining grammars. In *ACL 1985*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines. In *IWPT 2003*.

Automatic Question Generation for Vocabulary Assessment

Jonathan C. Brown
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
jonbrown@cs.cmu.edu

Gwen A. Frishkoff
Learning Research &
Development Center
University of Pittsburgh
Pittsburgh, PA 15260 USA
gwenf@pitt.edu

Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
max@cs.cmu.edu

Abstract

In the REAP system, users are automatically provided with texts to read targeted to their individual reading levels. To find appropriate texts, the user's vocabulary knowledge must be assessed. We describe an approach to automatically generating questions for vocabulary assessment. Traditionally, these assessments have been hand-written. Using data from WordNet, we generate 6 types of vocabulary questions. They can have several forms, including wordbank and multiple-choice. We present experimental results that suggest that these automatically-generated questions give a measure of vocabulary skill that correlates well with subject performance on independently developed human-written questions. In addition, strong correlations with standardized vocabulary tests point to the validity of our approach to automatic assessment of word knowledge.

1 Introduction

The REAP system automatically provides users with individualized authentic texts to read. These texts, usually retrieved from the Web, are chosen to satisfy several criteria. First, they are selected to match the reading level of the student (Collins-Thompson and Callan, 2004). They must also have vocabulary terms known to the student. To meet this goal, it is necessary to construct an accurate model of the student's vocabulary knowledge (Brown and Eskenazi, 2004). Using this model, the system can locate documents that include a given percentage (e.g., 95%) of words that are known to the student. The remaining percentage (e.g. 5%) consists of new words that the student needs to learn. This percentage is controlled so that there is not so much stretch in the document that the student cannot focus their attention on understanding the new words and the meaning of the text. After reading the text, the student's understanding of new words is assessed. The student's responses are used to update the student model, to support retrieval of furture documents that take into account the changes in student word knowledge.

In this paper, we describe our work on automatic generation of vocabulary assessment questions. We also report results from a study that was designed to assess the validity of the generated questions. In addition to the importance of these assessments in the REAP system, tests of word knowledge are central to research on reading and language and are of practical importance for student placement and in enabling teachers to track improvements in word knowledge throughout the school year. Because tests such as these are traditionally hand-written, development is time-consuming and often relies on methods that are informal and subjective. The research described here addresses these issues through development of automated, explicit methods for generation of vocabulary tests. In addition, these tools are designed to capture the graded and complex nature of word knowledge, allowing for more fine-grained assessment of word learning.

2 Measuring Vocabulary Knowledge

Word knowledge is not all-or-none. Rather, there are different aspects, such as knowledge of the spoken form, the written form, grammatical behav-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 819–826, Vancouver, October 2005. ©2005 Association for Computational Linguistics

ior, collocation behavior, word frequency, stylistic register constraints, conceptual meaning, and the associations a word has with other related words (Nation, 1990). In this paper, we focus on knowledge of conceptual word meaning. Because word meaning itself is complex, our focus is not simply on all-or-none estimates of vocabulary knowledge, but also on graded and incomplete knowledge of meanings that readers possess for different words and at different stages of acquisition.

Several models have been proposed to account for these multiple levels of word knowledge. For example, Dale posited four stages of knowledge of word meaning (Dale and O'Rourke, 1965). In stage 1, the subject has never seen the word. In stage 2, she has seen the word but is unable to verbalize its meaning. In stage 3, the subject recognizes the word in a given context and has partial word knowledge. In stage 4, the subject has full word knowledge, and can explain the word meaning so that its usage is clear in multiple contexts.

Stahl (1986) proposed a similar model of word knowledge, the levels of which overlap with Dale's last two stages. According to this model, the first level is characterized by association processing, or the passive association of the new word meaning with other, familiar concepts. The second level, comprehension processing, involves active comprehension of the word in a particular context. The third level, generation processing, requires usage of a word in a novel context reflecting a deep (and multidimensional) understanding of its meaning.

Taking Stahl's framework as a working model, we constructed multiple types of vocabulary questions designed to assess different "stages" or "levels" of word knowledge.

3 Question Generation

In this section, we describe the process used to generate vocabulary questions. After introducing the WordNet resource we discuss the six question types and the forms in which they appear. The use of distractors is covered in section 3.3.

3.1 WordNet

WordNet is a lexical resource in which English nouns, verbs, adjectives, and adverbs are grouped into synonym sets. A word may appear in a number of these synonym sets, or synsets, each corresponding to a single lexical concept and a single sense of the word (Fellbaum ed., 1998). The word "bat" has ten distinct senses and thus appears in ten synsets in WordNet. Five of these senses correspond to noun senses, and the other five correspond to verb senses. The synset for the verb sense of the word which refers to batting one's eyelashes contains the words "bat" and "flutter", while the synset for the noun sense of the word which refers to the flying mammal contains the words "bat" and "chiropteran". Each sense or synset is accompanied by a definition and, often, example sentences or phrases. A synset can also be linked to other synsets with various relations, including synonym, antonym, hypernym, hyponym, and other syntactic and semantic relations (Fellbaum ed., 1998). For a particular word sense, we programmatically access WordNet to find definitions, example phrases, etc.

3.2 Question Types

Given Stahl's three levels of word mastery and the information available in WordNet, we generated 6 types of questions: definition, synonym, antonym, hypernym, hyponym, and cloze questions.

In order to retrieve data from WordNet, we must choose the correct sense of the word. The system can work with input of varying specificity. The most specific case is when we have all the data: the word itself and a number indicating the sense of the word with respect to WordNet's synsets. When the target words are known beforehand and the word list is short enough, the intended sense can be hand-annotated. More often, however, the input is comprised of just the target word and its part of speech (POS). It is much easier to annotate POS than it is to annotate the sense. In addition, POS tagging can be done automatically in many cases. In the REAP system, where the user has just read a specific text, the words of the document were already automatically POS annotated. When there is only one sense of the word per part of speech, we can simply select the correct sense of the word in WordNet. Otherwise, we select the most frequently used sense of the word with the correct POS, using WordNet's frequency data. If we have only the word, we select the most frequent sense, ignoring part of speech. Future work will use word sense disambiguation techniques to automatically determine the correct word sense given a document that includes the target word, as in REAP (Brown and Eskenazi, 2004).

Once the system has determined the word sense, it can retrieve data from WordNet for each of the 6 question types. The definition question requires a definition of the word, available in WordNet's gloss for the chosen sense. The system chooses the first definition which does not include the target word. This question should provide evidence for the first of Stahl's three levels, association processing, although this was not explicitly evaluated.

The synonym question has the testee match the target word to a synonym. The system can extract this synonym from WordNet using two methods. One method is to select words that belong to the same synset as the target word and are thus synonyms. In addition, the synonym relation in WordNet may connect this synset to another synset, and all the words in the latter are acceptable synonyms. The system prefers words in the synset to those in synonym synsets. It also restricts synonyms to single words and to words which are not morphological variants of the target word. When more than one word satisfies all criteria, the most frequently used synonym is chosen, since this should make the question easier. This question could be considered either association processing or comprehension processing. If the testee has seen this synonym (e.g. as a hint), this question type would require association processing as a word is simply being associated with another already-presented word. Otherwise, this may require comprehension processing – understanding beyond memorization.

The antonym question requires matching a word with an antonymous word. WordNet provides two kinds of relations that can be used to procure antonyms: direct and indirect antonyms. Direct antonyms are antonyms of the target word, whereas indirect antonyms are direct antonyms of a synonym of the target. The words "fast" and "slow" are direct antonyms of one another. The word "quick" does not have a direct antonym, but it does have an indirect antonym, "slow", via "fast", its synonym. When more than one antonym is available, the most frequently used is chosen. Unless the testee has already seen the antonym, this type of question is normally considered to provide evidence for Stahl's second level, comprehension processing.

The hypernym and hyponym questions are similar in structure. Hypernym is the generic term used to describe a whole class of specific instances. The word "organism" is a hypernym of "person". Hyponyms are members of a class. The words "adult", "expert" and "worker" are hyponyms of "person". For the questions the testee matches the target word to either a hypernym or hyponym. For more than one possibility, the most frequently used term is chosen. Unless the testee has previously seen the hypernym or hyponym, these questions are normally regarded as providing evidence for Stahl's second level.

Cloze is the final question type. It requires the use of the target word in a specific context, either a complete sentence or a phrase. The example sentence or phrase is retrieved from the gloss for a specific word sense in WordNet. There is often more than one example phrase. The system prefers longer phrases, a feature designed to increase the probability of retrieving complete sentences. Passages using the target word are preferred, although examples for any of the words in the synset are appropriate. The present word is replaced by a blank in the cloze question phrase. Some consider a cloze question to be more difficult than any of the other question types, but it is still expected to provide evidence for Stahl's second level.

Although our question types provide evidence for the highest level of schemes such as Dale's four stages, they do not provide evidence for Stahl's highest level, generation processing, where the testee must, for instance, write a sentence using the word in a personalized context. We expect questions that provide evidence of this level to require free-form or near-free-form responses, which we do not yet allow. We expect the six question types to be of increasing difficulty, with definition or synonym being the easiest and cloze the hardest.

3.3 Question Forms

Each of the 6 types of questions can be generated in several forms, the primary ones being wordbank and multiple-choice. In wordbank, the testee sees a list of answer choices, followed by a set of questions or statements (see Figure 1). For the definition version, each of the items below the wordbank is a definition. The testee must select the word which best corresponds to the definition. For the synonym and antonym questions, the testee selects the word which is the most similar or the most opposite in meaning to the synonym or antonym. For the hypernym and hyponym question types, the testee is asked to complete phrases such as "___ is a kind of person" (with target "adult") or "person is a kind of ___" (with target "organism"). In the cloze question, the testee fills in the blank with the appropriate word. There is traditionally one question for each target word in the wordbank. These questions require no information beyond the target words and their definitions, synonyms, hypernyms, etc.

Wordbank:

verbose   infallible   obdurate   opaque

Choose the word from the wordbank that best completes each phrase below:

1. ___ windows of the jail
2. the Catholic Church considers the Pope ___
3. ___ and ineffective instructional methods
4. the child's misery would move even the most ___ heart

Fig. 1. Example Wordbank Question

The second generated form is multiple-choice, with one question per target word. The testee sees the main question, the stem, followed by several answer choices, of which only one is correct (see Figure 2). Depending on the question type, the target word may appear in either the stem or the answer choices. For the definition question type, the stem holds the definition of the target word and one of the answer choices is the target word. For the word "verbose", the stem would be "using or containing too many words" and the choices "ancillary", "churlish", "verbose", and "convivial". The cloze question is of a similar form, with the stem containing the example sentence or phrase with a blank where the target word should be used. For "verbose", we have the stem "___ and ineffective instructional methods" and choices "verbose", "incipient", "invidious", and "titular". For the synonym, antonym, hypernym, and hyponym questions, the target word appears in the stem instead of the answer choices. The synonym question for the word "verbose" would have the stem "Select the word that is most similar in meaning to the word verbose" with choices "inflammable", "piping", matrilineal", and "long-winded". The antonym question would have the stem "Select the word that is most opposite in meaning to the word verbose" and the choices "discernable", "concise", "unbroken", and "soused". Figure 2 shows a formatted example of an automatically generated multiple-choice cloze question for the word "obdurate".

Choose the word that best completes the phrase below:

the child's misery would move even the most ___ heart

A) torpid
B) invidious
C) stolid
D) obdurate

Fig. 2. Example Multiple-Choice Cloze Question

Two issues to consider when creating multiple-choice format questions are the wording or appearance of the questions and the criteria for selection of distractors. We followed the guidelines for good multiple-choice questions described by researchers such as Graesser and Wisher (2001). In accord with these guidelines, our questions had 4 choices, although the number of choices is a variable supplied to the question generation software. We also considered the most appropriate wording for these questions, leading us to choose stems such as "Select the word that is most similar in meaning to the word *plausible*" for the synonym question rather than "Choose the word that means the same as the word *plausible*." The latter would be problematic when the correct answer is a near-synonym rather than a word with precisely the same meaning.

Concerning distractor choice, the question generation system chooses distractors of the same part of speech and similar frequency to the correct answer, as recommended by Coniam (1997). For the synonym, antonym, hypernym, and hyponym questions, the correct answer is the highest frequency word of *all* the words chosen from WordNet that satisfy all the criteria. Thus, the distractors are of the same POS and similar frequency to the synonym, antonym, or whatever word is the correct answer, as opposed to the target word. The system chooses distractors from Kilgarriff's (1995) word frequency database, based on the British National Corpus (BNC) (Burnage, 1991). The system chooses 20 words from this database that are of the same POS and are equal or similar in frequency to the correct answer, and randomly chooses the distractors from these words. Since the distractors may be different for each run of the question generation software, slightly different versions of the same basic question may appear. The words of the BNC and the word frequency database have been POS tagged using the CLAWS tagger (Leech, 1994). This tagger uses detailed POS tags, enabling us to choose distractors that are, for instance, verbs in the past tense, when the correct answer is such as verb, instead of selecting verbs of unknown tense. In the definition and cloze questions, the correct answer is the target word itself, so distractors are chosen based on this word. The system also restricts distractors to be in the list of target words so that the testee cannot simply choose the word that appears in the stems of other questions.

An alternate multiple-choice question format is used when the testee has just read a document using the target word, as in the REAP system (Brown and Eskenazi, 2004). In this case, the system also attempts to finds words which may be semantically related to the correct answer, as in (Nagy, 1985). This is done by choosing distractors that satisfy the standard criteria and were present in the document. This should increase the chance that the distractors are semantically related and eliminate the chance that a testee will simply select as the correct answer the word that appeared in the document they just read, without understanding the word meaning.

4 Question Assessment

The validity of the automatically generated vocabulary questions was examined in reference to human-generated questions for 75 low-frequency English words. We compared student performance (accuracy and response time) on the computer and human-generated questions. We focused on the automatically generated multiple-choice questions, with distractors based on frequency and POS. We did not examine using more complicated strategies for picking distractors or assume there was an associated text. Four of the six computer-generated question types were assessed: the definition, synonym, antonym, and cloze questions. Hypernym and hyponym questions were excluded, since we were unable to generate a large number of these questions for adjectives, which constitute a large portion of the word list. Subject scores on the computer and human-generated assessments were compared with scores on standardized measures of reading and vocabulary skill, as described below.

4.1 Question Coverage

Potential experimental stimuli comprised 156 low-frequency and rare English words that have been used in previous studies of vocabulary skill in native English-speaking adults. We first examined the percentage of words for which we could generate various question types. We were unable to generate any questions for 16 of these words, or ~9% of the list, since they were not in WordNet. Table 1 shows the percentage of words for which each of the four question types was generated. All four questions were able to be generated for only 75 (about half) of the words. Therefore, the experimental word list included only these 75 items. Given the rarity of the words, we predicted that the percentage of words for which we could generate questions would be lower than average. However, we expected that the percentage of words for which we could generate synonym and antonym questions to be higher than average, due to the heavy focus on adjectives in this list.

Question type	Percentage of Questions Generated
Definition Question	91%
Synonym Question	80%
Antonym Question	60%
Cloze Question	60%

Table 1. Question Coverage for the 156-Word List

4.2 Experiment Design

Behavioral measures of vocabulary knowledge were acquired for the 75 target words using the four computer-generated question types described above, as well as five human-generated question types. The human-generated questions were developed by a group of three learning researchers, without knowledge of the computer-generated question types. Researchers were asked merely to develop a set of question types that could be used to assess different levels, or different aspects, of word knowledge. Examples of each question type (including distractors) were hand-written for each of the 75 words.

Two of the five human-generated assessments, the synonym and cloze questions, were similar in form to the corresponding computer-generated question types in that they had the same type of stem and answer. The other three human-generated questions included an inference task, a sentence completion task, and a question based on the Osgood semantic differential (Osgood, 1970). In the inference task, participants were asked to select a context where the target word could be meaningfully applied. For example, the correct response to the question "Which of the following is most likely to be lenitive?" was "a glass of iced tea," and distractors were "a shot of tequila," "a bowl of rice," and "a cup of chowder." In the sentence completion task, the participant was presented with a sentence fragment containing the target word and was asked to choose the most probable completion. For example, the stem could be "The music was so lenitive…," with the correct answer "…it was tempting to lie back and go to sleep," and with distractors such as "…it took some concentration to appreciate the complexity." The fifth question type was based on the Osgood semantic differential, a factor-analytic model of word-level semantic dimensions (Osgood, 1970). Numerous studies using the Osgood paradigm have shown that variability in the semantic "structure" of word meanings can largely be accounted for in terms of three dimensions, valence (good–bad), potency (strong–weak), and activity (active–passive). In our version of the Osgood task, subjects were asked to classify a word such as "lenitive" along one of these dimensions (e.g., more good or more bad).

In addition to the human-generated questions, we administered a battery of standardized tests, including the Nelson-Denny Reading Test, the Raven's Matrices Test, and the Lexical Knowledge Battery. The Nelson-Denny Reading Test is a standardized test of vocabulary and reading comprehension (Brown, 1981). The Raven's Matrices Test is a test of non-verbal reasoning (Raven, 1960). The Lexical Knowledge Battery has multiple subsections that test orthographic and phonological skills (Perfetti and Hart, 2001).

Twenty-one native-English speaking adults participated in two experiment sessions. Session 1 lasted for about one hour and included the battery of vocabulary and reading-related assessments described above. Session 2 lasted between two and three hours and comprised 10 tasks, including the five human and four computer-generated questions. The experiment began with a confidence-rating task, in which participants indicated with a key press how well they knew the meaning of each target word (on a 1–5 scale). This task was not speeded. For the remaining tasks, subjects were asked to respond "as quickly as possible without making errors." Test items for a given question type were answered together. The order of the tasks (question types) and the order of the 75 items within each task were randomized across subjects.

4.3 Experiment Results

We report on four aspects of this study: participant performance on questions, correlations between question types, correlations with confidence ratings, and correlations with external assessments.

Mean accuracy scores for each question type varied from .5286 to .6452. Performance on individual words and across subjects (averaging across words) varied widely. The easiest question types (those with the highest average accuracy), were the computer-generated definition task and the human-generated semantic differential task, both having mean accuracy scores of .6452. The hardest was the computer-generated cloze task, with a mean score of .5286. The accuracy on computer-generated synonym and antonym questions fall between these two limits, with slightly greater accuracy on the synonym type. This implies a general ordering of difficulty from definition to cloze, as expected. The accuracies on the other human-generated questions also fall into this range.

We also computed correlations between the different question types. Mean accuracies were highly and statistically significantly correlated across the nine question types ($r>.7$, $p<.01$ for all correlations). The correlation between participant accuracy on the computer-generated synonym and the human-generated synonym questions was particularly high ($r=.906$), as was the correlation between the human and computer cloze questions ($r= .860$). The pattern of correlations for the response-time (RT) data was more complicated and is discussed elsewhere (Frishkoff et al, In Prep). Importantly, RTs for the human versus computer versions of both the synonym and cloze questions were strongly correlated ($r>.7$, $p<.01$), just as for the accuracy results. The accuracy correlations imply that the computer-generated questions are giving a measure of vocabulary skill for specific words that correlates well with that of the human-generated questions.

An item analysis (test item discrimination) was also performed. For each word, scores on a particular question type were compared with the composite test score for that word. This analysis revealed relatively low correlations ($.12 < r < .25$) between the individual question types and the test as a whole (without that question type). Since the question types were designed to test different aspects of vocabulary knowledge, this result is encouraging.

In addition, the average total-score correlations for the four computer-generated questions (r=.18) and for the five human-generated questions (r=.19) were not significantly different. This is positive, since it suggests that the human and computer-generated vocabulary test are accounting for similar patterns of variance across the different question types.

The average correlation between accuracy on the question types and confidence ratings for a particular word was .265. This correlation was unexpectedly low. This may be because participants thought they knew these words, but were confused by their rarity, or because confidence simply does not correlate well with accuracy. Further work is needed to determine whether confidence ratings can be accurate predictors of vocabulary knowledge.

Finally, we examined correlations between participant performance on the nine question types and the external assessments. The correlations between the accuracy on each of the nine question types and the Nelson-Denny vocabulary subtest were fairly high (.61 < r < .85, p=.01 for all comparisons). Thus, both the computer and human-generated questions show good correspondence with an external assessment of vocabulary skill. Correlations between the accuracy on the question types and the Nelson-Denny reading comprehension test were mixed, showing a higher correlation with vocabulary than reading comprehension. Correlations between the accuracy on the nine question types and the Raven's Matrices test of nonverbal reasoning were positive, but low and not statistically significant. This provides strong evidence that the computer-generated vocabulary questions tap vocabulary knowledge specifically, rather than intelligence in general.

5 Related Work

Cloze tests are one area of related work. They were originally intended to measure text readability (Taylor, 1953) since native speakers should be able to reproduce certain removed words in a readable text. Other researchers have used it to assess reading comprehension (Ruddell, 1964), with students filling in the blanks, given a high quality text. The main issue in automating the creation of cloze tests is determining which words to remove from the text. Coniam (1997) examined a several options for determining the words to remove and produced relatively good-quality cloze tests by removing words with the same POS or similar frequency.

Wolfe (1976) automatically generated reading comprehension questions. This involved various techniques for rewriting sentences into questions, testing syntactic understanding of individual sentences. Of the 50 questions Wolfe was able to generate for a single text, 34 were found to be satisfactory. More recently, Kunichika (2003) carried out work in automatically generating reading comprehension questions that included both syntactic and semantic questions, and was able to generate several different types of questions, including asking about the content of a sentence, using dictionaries of synonyms and antonyms to generate questions such as "Is Jane busy?" from sentences like "Jane is free.", and testing semantic understanding across sentence boundaries. Approx. 93% of the generated questions were found to be satisfactory.

Aist (2001) automatically generated factoids to assist students reading. The factoids gave a synonym, an antonym, or a hypernym for the word, which were automatically extracted from WordNet. He also automated the creation of a single type of vocabulary question, with the target word in the stem and the correct answer a synonym, hypernym, or sibling from WordNet. It is unclear what type of vocabulary knowledge this question would tap, given the different possible answers.

6 Conclusions

Extending our experiments to the question types that we have not yet assessed is an important next step. In addition, we want to assess questions individually, evaluating their use of distractors. Finally, we need to assess questions generated on word lists with different characteristics.

There are also a number of ongoing extensions to this project. One is the creation of new question types to test other aspects of word knowledge. Another is using other resources such as text collections to enable us to generate more questions per word, especially for the cloze questions. In addition, we are looking at ways to predict word knowledge using confidence ratings and morphological and semantic cohorts in situations where we cannot perform a standard assessment or cannot test all the vocabulary words we would like to.

In this paper, we have described our work in automatically generating questions for vocabulary assessment. We have described the six types of computer-generated questions and the forms in which they appear. Finally, we have presented evidence that the computer-generated questions give a measure of vocabulary skill for individual words that correlates well with human-written questions and standardized assessments of vocabulary skill.

Acknowledgements

The authors would like to thank Jamie Callan and Kevyn Collins-Thompson for their help in this research. The authors would also like to thank Eve Landen, Erika Taylor, and Charles Perfetti for their assistance with experimental stimuli and data collection. This project is supported U.S. Department of Education, Institute of Education Sciences, Award #R305G030123, and the APA/IES Postdoctoral Education Research Training fellowship awarded by the Department of Education, Institute of Education Sciences, Grant #R305U030004. 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 U.S. Department of Education.

References

Gregory Aist. 2001. Towards automatic glossarization: automatically constructing and administering vocabulary assistance factoids and multiple-choice assessment, *International Journal of AI in Ed.*, 2001.

James Brown, J. M. Bennett, and Gerald Hanna. 1981. *The Nelson-Denny Reading Test*. Chicago: The Riverside Publishing Company.

Jonathan Brown and Maxine Eskenazi. 2004. Retrieval of Authentic Documents for Reader-Specific Lexical Practice. In *Proceedings of InSTIL/ICALL Symposium 2004*. Venice, Italy, 2004.

Gavin Burnage. 1991. Text Corpora and the British National Corpus. *Computers & Texts* 2, Nov, 1991.

Kevyn Collins-Thompson and Jamie Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the HLT/NAACL 2004 Conference*. Boston, 2004.

David Coniam. 1997. A preliminary inquiry into using corpus word frequency data in the automatic generation of English language cloze tests. *CALICO Journal*, Volume 14, No. 2.

Edgar Dale and Joseph O'Rourke. 1986. *Vocabulary building*. Columbus, Ohio: Zaner-Bloser.

Christiane Fellbaum, Ed. 1998. *WordNet. An electronic lexical database*. Ed. by Christiane Fellbaum, preface by George Miller. Cambridge, MA: MIT Press; 1998.

Arthur C. Graesser, R. A. Wisher. 2001. *Question Generation as a Learning Multiplier in Distributed Learning Environments*. Army research inst for the behavioral and social sciences Alexandria VA. Report number A654993, 2001.

Adam Kilgarriff. 1995. http://www.itri.brighton.ac.uk/~Adam.Kilgarriff/bnc-readme.html

H. Kunichika, T. Katayama, T. Hirashima, and A. Takeuchi. 2003. Automated question generation methods for intelligent English learning systems and its evaluation. *Proceedings of ICCE2004*, Hong Kong.

G. Leech, R. Garside, and M. Bryant. 1994. CLAWS4: The tagging of the British National Corpus. In *Proc. of 15th International Conference on Computational Linguistics*, Kyoto, Japan, 622-628, 1994.

W.E. Nagy, P.A. Herman, and R.C. Anderson. 1985. Learning words from context. *Reading Research Quarterly*, 20, 233-253.

Paul Nation. 1990. *Teaching and learning vocabulary*. Rowley, MA: Newbury House.

Charles E. Osgood, P. H. Tannenbaum, and G. J. Suci. 1957. *The Measurement of Meaning*. Urbana: University of Illinois Press.

Charles A. Perfetti, and Lesley Hart. 2001. The lexical quality hypothesis. In L. Verhoeven, C. Elbro & P. Reitsma (Eds.), *Precursors of Functional Literacy* (Vol. 11, pp. 67–86). Amsterdam: John Benjamins.

J.C. Raven. 1960. *Progressive matrices, standard*. San Antonio, TX: Psychological Corporation.

R. B. Ruddell. 1964. A study of the cloze comprehension technique in relation to structurally controlled reading material. *Improvement of Reading Through Classroom Practice*, 9, 298-303.

Steven A. Stahl. 1986. Three principals of effective vocabulary instruction. *Journal of Reading*, 29.

W.L. Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism Quarterly*, 30.

John H. Wolfe. 1976. Automatic question generation from text - an aid to independent study. *ACM SIGCUE Bulletin*, 2(1), 104-112.

Parallelism in Coordination as an Instance of Syntactic Priming: Evidence from Corpus-based Modeling

Amit Dubey and **Patrick Sturt** and **Frank Keller**
Human Communication Research Centre, Universities of Edinburgh and Glasgow
2 Buccleuch Place, Edinburgh EH8 9LW, UK
{adubey,sturt,keller}@inf.ed.ac.uk

Abstract

Experimental research in psycholinguistics has demonstrated a parallelism effect in coordination: speakers are faster at processing the second conjunct of a coordinate structure if it has the same internal structure as the first conjunct. We show that this phenomenon can be explained by the prevalence of parallel structures in corpus data. We demonstrate that parallelism is not limited to coordination, but also applies to arbitrary syntactic configurations, and even to documents. This indicates that the parallelism effect is an instance of a general syntactic priming mechanism in human language processing.

1 Introduction

Experimental work in psycholinguistics has provided evidence for the so-called parallelism preference effect: speakers processes coordinated structures more quickly when the two conjuncts have the same internal syntactic structure. The processing advantage for parallel structures has been demonstrated for a range coordinate constructions, including NP coordination (Frazier et al., 2000), sentence coordination (Frazier et al., 1984), and gapping and ellipsis (Carlson, 2002; Mauner et al., 1995).

The parallelism preference in NP coordination can be illustrated using Frazier et al.'s (2000) Experiment 3, which recorded subjects' eye-movements while they read sentences like (1):

(1) a. Terry wrote a long novel and <u>a short poem</u> during her sabbatical.
 b. Terry wrote a novel and <u>a short poem</u> during her sabbatical

Total reading times for the underlined region were faster in (1-a), where *short poem* is coordinated with a syntactically parallel noun phrase (*a long novel*), compared to (1-b), where it is coordinated with a syntactically non-parallel phrase.

These results raise an important question that the present paper tries to answer through corpus-based modeling studies: what is the mechanism underlying the parallelism preference? One hypothesis is that the effect is caused by low-level processes such as *syntactic priming*, i.e., the tendency to repeat syntactic structures (e.g., Bock, 1986). Priming is a very general mechanism that can affect a wide range of linguistic units, including words, constituents, and semantic concepts. If the parallelism effect is an instance of syntactic priming, then we expect it to apply to a wide range of syntactic construction, and both within and between sentences. Previous work has demonstrated priming effects in corpora (Gries, 2005; Szmrecsanyi, 2005); however, these results are limited to instances of priming that involve a choice between two structural alternatives (e.g., dative alternation). In order to study the parallelism effect, we need to model priming as general syntactic repetition (independent of the structural choices available). This is what the present paper attempts.

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 second conjunct is parallel to the first one. If the copying account is correct, then we would expect parallelism effects to be restricted to coordinate structures and would not apply in other contexts.

In the present paper, we present corpus evidence that allows us to distinguish between these two competing explanations. Our investigation will proceed as follows: we first establish that there is evidence for a parallelism effect in corpus data (Section 3). This is a crucial prerequisite for our wider investigation: previous work has only dealt with parallelism in comprehension, hence we need to establish that parallelism is also present in production data, such as corpus data. We then investigate whether the parallelism effect is restricted to coordination, or whether it also applies also arbitrary syntactic configurations. We also test if parallelism can be found for larger segments of text, including, in the limit, the whole document (Section 4). Then we investigate parallelism in dialog, testing the psycholinguistic prediction that parallelism in dialog occurs between speakers (Section 5). In the next section, we discuss a number of methodological issues and explain the way we measure parallelism in corpus data.

2 Adaptation

Psycholinguistic studies have shown that priming affects both speech production (Bock, 1986) and comprehension (Branigan et al., 2005). The importance of comprehension priming has also been noted by the speech recognition community (Kuhn and de Mori, 1990), who use so-called caching language models to improve the performance of speech comprehension software. The concept of caching language models is quite simple: a cache of recently seen words is maintained, and the probability of words in the cache is higher than those outside the cache.

While the performance of caching language models is judged by their success in improving speech recognition accuracy, it is also possible to use an abstract measure to diagnose their efficacy more closely. Church (2000) introduces such a diagnostic for lexical priming: adaptation probabilities. Adaptation probabilities provide a method to separate the general problem of priming from a particular implementation (i.e., caching models). They measure the amount of priming that occurs for a given construction, and therefore provide an upper limit for the performance of models such as caching models.

Adaptation is based upon three concepts. First is the *prior*, which serves as a baseline. The prior measures the probability of a word appearing, ignoring the presence or absence of a prime. Second is the *positive adaptation*, which is the probability of a word appearing given that it has been primed. Third is the *negative adaptation*, the probability of a word appearing given it has not been primed.

In Church's case, the prior and adaptation probabilities are estimated as follows. If a corpus is divided into individual documents, then each document is then split in half. We refer to the halves as the prime set (or prime half) and the target set (or target half).[1] We measure how frequently a document half contains a particular word. For each word w, there are four combinations of the prime and target halves containing the word. This gives us four frequencies to measure, which are summarized in the following table:

$f_{w_{p,t}}$	$f_{w_{\bar{p},t}}$
$f_{w_{p,\bar{t}}}$	$f_{w_{\bar{p},\bar{t}}}$

These frequencies represent:

$f_{w_{p,t}}$	=	# of times w occurs in prime set and target set
$f_{w_{\bar{p},t}}$	=	# of times w occurs in target set but not prime set
$f_{w_{p,\bar{t}}}$	=	# of times w occurs in prime set but not target set
$f_{w_{\bar{p},\bar{t}}}$	=	# of times w does not occur in either target set or prime set

In addition, let N represent the sum of these four frequencies. From the frequencies, we may formally define the prior, positive adaptation and negative adaptation:

(1) **Prior** $$P_{\text{prior}}(w) = \frac{f_{w_{p,t}} + f_{w_{\bar{p},t}}}{N}$$

(2) **Positive Adaptation** $$P_{+}(w) = \frac{f_{w_{p,t}}}{f_{w_{p,t}} + f_{w_{p,\bar{t}}}}$$

(3) **Negative Adaptation** $$P_{-}(w) = \frac{f_{w_{\bar{p},t}}}{f_{w_{\bar{p},t}} + f_{w_{\bar{p},\bar{t}}}}$$

In the case of lexical priming, Church observes that $P_{+} \gg P_{\text{prior}} > P_{-}$. In fact, even in cases when P_{prior} quite small, P_{+} may be higher than 0.8. Intuitively, a positive adaptation which is higher than the prior entails that a word is likely to reappear in the target set given that it has already appeared in the prime set. We intend to show that adaptation probabilities provide evidence that syntactic constructions behave

[1]Our terminology differs from that of Church, who uses 'history' to describe the first half, and 'test' to describe the second. Our terms avoid the ambiguity of the phrase 'test set' and coincide with the common usage in the psycholinguistic literature.

similarity to lexical priming, showing positive adaptation P_+ greater than the prior. As P_- must become smaller than P_{prior} whenever P_+ is larger than P_{prior}, we only report the positive adaptation P_+ and the prior P_{prior}.

While Church's technique was developed with speech recognition in mind, we will show that it is useful for investigating psycholinguistic phenomenon. However, the connection between cognitive phenomenon and engineering approaches go in both directions: it is possible that syntactic parsers could be improved using a model of syntactic priming, just as speech recognition has been improved using models of lexical priming.

3 Experiment 1: Parallelism in Coordination

In this section, we investigate the use of Church's adaptation metrics to measure the effect of syntactic parallelism in coordinated constructions. For the sake of comparison, we restrict our study to several constructions used in Frazier et al. (2000). All of these constructions occur in NPs with two coordinate sisters, i.e., constructions such as NP_1 CC NP_2, where CC represents a coordinator such as *and*.

3.1 Method

The application of the adaptation metric is straightforward: we pick NP_1 as the prime set and NP_2 as the target set. Instead of measuring the frequency of lexical elements, we measure the frequency of the following syntactic constructions:

SBAR An NP with a relative clause, i.e., NP $\rightarrow$ NP SBAR.

PP An NP with a PP modifier, i.e., NP $\rightarrow$ NP PP.

NN An NP with a single noun, i.e., NP $\rightarrow$ NN.

DT NN An NP with a determiner and a noun, i.e., NP $\rightarrow$ DT NN.

DT JJ NN An NP with a determiner, an adjective and a noun, i.e., NP $\rightarrow$ DT JJ NN.

Parameter estimation is accomplished by iterating through the corpus for applications of the rule NP $\rightarrow$ NP CC NP. From each rule application, we create a list of prime-target pairs. We then estimate adaptation probabilities for each construction, by counting the number of prime-target pairs in which the construction does or does not occur. This is done similarly to the document half case described above. There are four frequencies of interest, but now they refer to the frequency that a particular construction (rather than a word) either occurs or does not occur in the prime and target set.

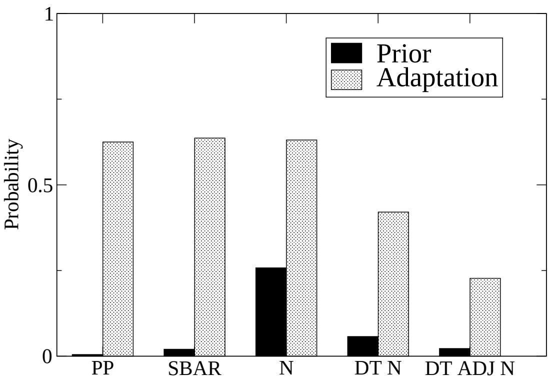


Figure 1: Adaptation within coordinate structures in the Brown corpus

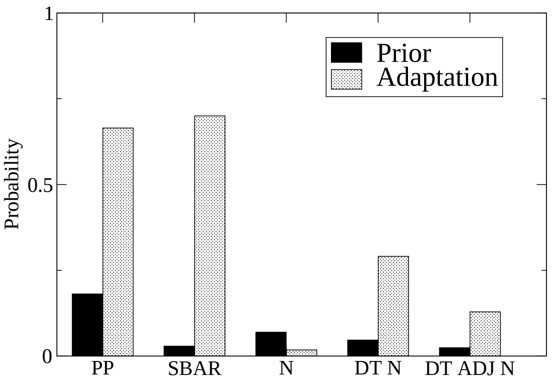


Figure 2: Adaptation within coordinate structures in the WSJ corpus

To ensure results were general across genres, we used all three parts of the English Penn Treebank: the Wall Street Journal (WSJ), the balanced Brown corpus of written text (Brown) and the Switchboard corpus of spontaneous dialog. In each case, we use the entire corpus.

Therefore, in total, we report 30 probabilities: the prior and positive adaptation for each of the five constructions in each of the three corpora. The primary objective is to observe the difference between the prior and positive adaptation for a given construction in a particular corpus. Therefore, we also perform a χ^2 test to determine if the difference between these two probabilities are statistically significant.

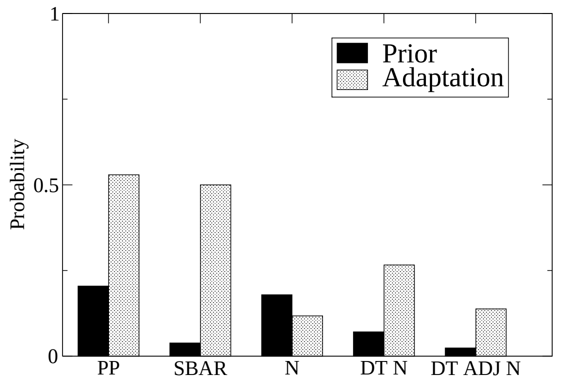


Figure 3: Adaptation within coordinate structures in the Switchboard corpus

3.2 Results

The results are shown in Figure 1 for the Brown corpus, Figure 2 for the WSJ and Figure 3 for Switchboard. Each figure shows the prior and positive adaptation for all five constructions: relative clauses (SBAR) a PP modifier (PP), a single common noun (N), a determiner and noun (DT N), and a determiner adjective and noun (DT ADJ N). Only in the case of a single common noun in the WSJ and Switchboard corpora is the prior probability higher than the positive adaptation. In all other cases, the probability of the given construction is more likely to occur in NP_2 given that it has occurred in NP_1. According to the χ^2 tests, all differences between priors and positive adaptations were significant at the 0.01 level. The size of the data sets means that even small differences in probability are statistically significant. All differences reported in the remainder of this paper are statistically significant; we omit the details of individual χ^2 tests.

3.3 Discussion

The main conclusion we draw is that the parallelism effect in corpora mirrors the ones found experimentally by Frazier et al. (2000), if we assume higher probabilities are correlated with easier human processing. This conclusion is important, as the experiments of Frazier et al. (2000) only provided evidence for parallelism in *comprehension* data. Corpus data, however, are *production* data, which means that the our findings are first ones to demonstrate parallelism effects in production.

The question of the relationship between comprehension and production data is an interesting one. We can expect that production data, such as corpus data, are generated by speakers through a process that involves self-monitoring. Written texts (such as the WSJ and Brown) involve proofreading and editing, i.e., explicit comprehension processes. Even the data in a spontaneous speech corpus such as Swtichboard can be expected to involve a certain amount of self-monitoring (speakers listen to themselves and correct themselves if necessary). It follows that it is not entirely unexpected that similar effects can be found in both comprehension and production data.

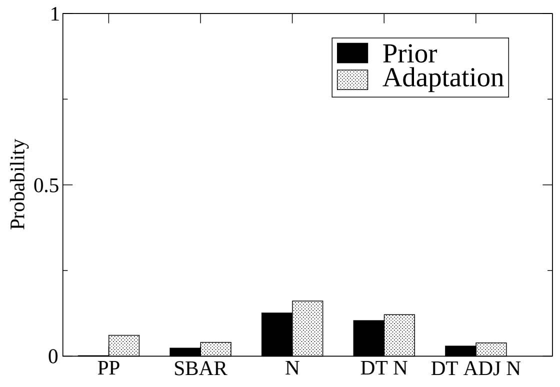


Figure 4: Adaptation within sentences in the Brown corpus

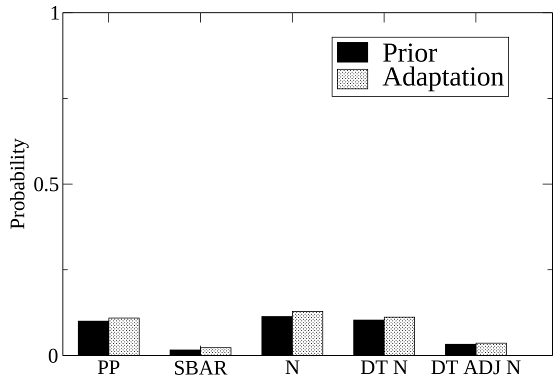


Figure 5: Adaptation within sentences in the WSJ corpus

4 Experiment 2: Parallelism in Documents

The results in the previous section showed that the parallelism effect, which so far had only been demonstrated in comprehension studies, is also attested in corpora, i.e., in production data. In the present experiment, we will investigate the mechanisms underlying the parallelism effect. As discussed in Section 1, there are two possible explana-

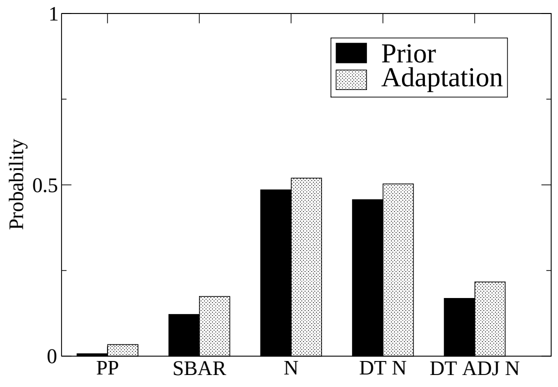


Figure 6: Adaptation between sentences in the Brown corpus

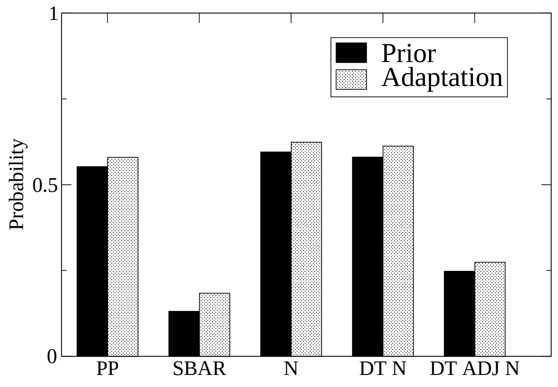


Figure 7: Adaptation between sentences in the WSJ corpus

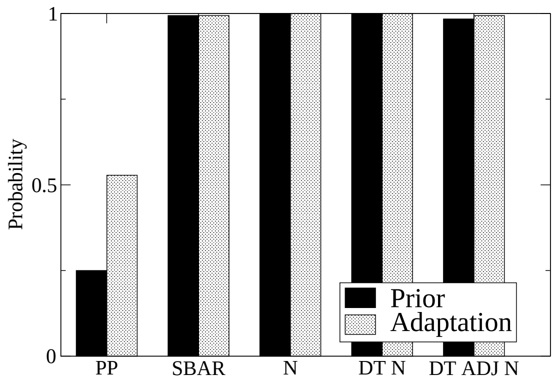


Figure 8: Adaptation within documents in the Brown corpus (all items exhibit weak yet statistically significant positive adaptation)

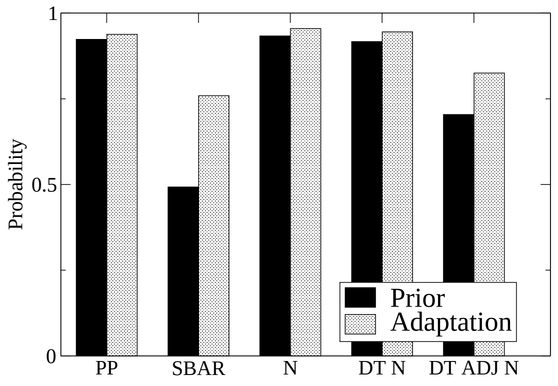


Figure 9: Adaptation within documents in the WSJ corpus

tion for the effect: one in terms of a construction-specific copying mechanism, and one in terms of a generalized syntactic priming mechanism. In the first case, we predict that the parallelism effect is restricted to coordinate structures, while in the second case, we expect that parallelism (a) is independent of coordination, and (b) occurs in the wider discourse, i.e., not only within sentences but also between sentences.

4.1 Method

The method used was the same as in Experiment 1 (see Section 3.1), with the exception that the prime set and the target set are no longer restricted to being the first and second conjunct in a coordinate structure. We investigated three levels of granularity: within sentences, between sentences, and within documents. *Within-sentence parallelism* occurs when the prime NP and the target NP occur within the same sentence, but stand in an arbitrary structural relationship. Coordinate NPs were excluded from this analysis, so as to make sure that any within-sentence parallelism is not confounded coordination parallelism as established in Experiment 1. *Between-sentence parallelism* was measured by regarding as the target the sentence immediately following the prime sentence. In order to investigate *within-document parallelism*, we split the documents into equal-sized halves; then the adaptation probability was computed by regarding the first half as the prime and the second half as the target (this method is the same as Church's method for measuring lexical adaptation).

The analyses were conducted using the Wall Street Journal and the Brown portion of the Penn Treebank. The document boundary was taken to be the file boundary in these corpora. The Switchboard corpus is a dialog corpus, and therefore needs to be treated differently: turns between speakers rather than sentences should be level of analysis. We will investigate this separately in Experiment 3 below.

4.2 Results

The results for the within-sentence analysis are graphed in Figures 4 and 5 for the Brown and WSJ corpus, respectively. We find that there is a parallelism effect in both corpora, for all the NP types investigated. Figures 6–9 show that the same is true also for the between-sentence and within-document analysis: parallelism effects are obtained for all NP types and for both corpora, even it the parallel structures occur in different sentences or in different document halves. (The within-document probabilities for the Brown corpus (in Figure 8) are close to one in most cases; the differences between the prior and adaptation are nevertheless significant.)

In general, note that the parallelism effects uncovered in this experiment are smaller than the effect demonstrated in Experiment 1: The differences between the prior probabilities and the adaptation probabilities (while significant) are markedly smaller than those uncovered for parallelism in coordinate structure.[2]

4.3 Discussion

This experiment demonstrated that the parallelism effect is not restricted to coordinate structures. Rather, we found that it holds across the board: for NPs that occur in the same sentence (and are not part of a coordinate structure), for NPs that occur in adjacent sentences, and for NPs that occur in different document halves. The between-sentence effect has been demonstrated in a more restricted from by Gries (2005) and Szmrecsanyi (2005), who investigate priming in corpora for cases of structural choice (e.g., between a dative object and a PP object for verbs like *give*). The present results extend this finding to arbitrary NPs, both within and between sentences.

The fact that parallelism is a pervasive phenomenon, rather than being limited to coordinate structures, strongly suggests that it is an instance of a general syntactic priming mechanism, which has been an established feature of accounts of the human sentence production system for a while (e.g., Bock, 1986). This runs counter to the claims made by Frazier et al. (2000) and Frazier and Clifton (2001), who have argued that parallelism only occurs in coordinate structures, and should be accounted for using a specialized copying mechanism. (It is important to bear in mind, however, that Frazier et al. only make explicit claims about comprehension, not about production.)

However, we also found that parallelism effects are clearly strongest in coordinate structures (compare the differences between prior and adaptation in Figures 1–3 with those in Figures 4–9). This could explain why Frazier et al.'s (2000) experiments failed to find a significant parallelism effect in non-coordinated structures: the effect is simply too week to detect (especially using the self-paced reading paradigm they employed).

5 Experiment 3: Parallelism in Spontaneous Dialog

Experiment 1 showed that parallelism effects can be found not only in written corpora, but also in the Switchboard corpus of spontaneous dialog. We did not include Switchboard in our analysis in Experiment 2, as this corpus has a different structure from the two text corpora we investigated: it is organized in terms of turns between two speakers. Here, we exploit this property and conduct a further experiment in which we compare parallelism effects between speakers and within speakers.

The phenomenon of structural repetition between speakers has been discussed in the experimental psycholinguistic literature (see Pickering and Garrod 2004 for a review). According to Pickering and Garrod (2004), the act of engaging in a dialog facilitates the use of similar representations at all linguistic levels, and these representations are shared between speech production and comprehension processes. Thus structural adaptation should be observed in a dialog setting, both within and between speakers. An alternative view is that production and comprehension processes are distinct. Bock and Loebell (1990) suggest that syntactic priming in speech production is due to facilitation of the retrieval and assembly procedures that occur during the formulation of utterances. Bock and Loebell point out that this production-based procedural view predicts a lack of priming between comprehension and production or vice versa, on the assumption that

[2]The differences between the priors and adaptation probabilities are also much smaller than noted by Church (2000). The probabilities of the rules we investigate have a higher marginal probability than the lexical items of interest to Church.

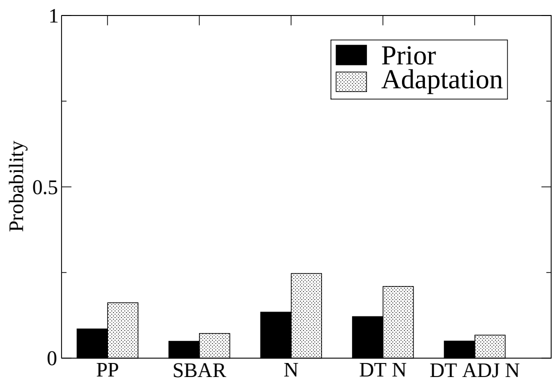


Figure 10: Adaptation between speakers in the Switchboard corpus

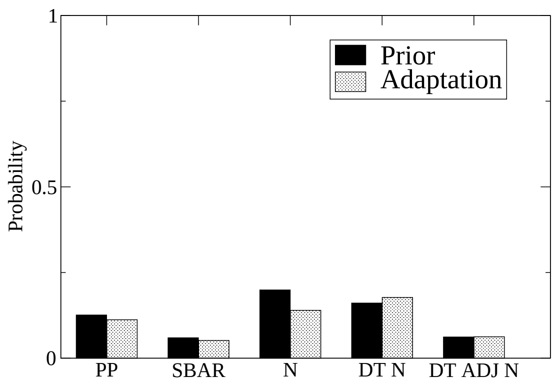


Figure 11: Adaptation within speakers in the Switchboard corpus

production and parsing use distinct mechanisms. In our terms, it predicts that between-speaker positive adaptation should not be found, because it can only result from priming from comprehension to production, or vice versa. Conversely, the prodedural view outlined by Bock and Loebell predicts that positive adaptation *should* be found within a given speaker's dialog turns, because such adaptation can indeed be the result of the facilitation of production routines within a given speaker.

5.1 Method

We created two sets of prime and target data to test within-speaker and between-speaker adaptation. The prime and target sets were defined in terms of pairs of utterances. To test between-speaker adaptation, we took each adjacent pair of utterances spoken by speaker A and speaker B, in each dialog, and these were treated as prime and target sets respectively. In the within-speaker analysis, the prime and target sets were taken from the dialog turns of only one speaker—we took each adjacent pair of dialog turns uttered by a given speaker, excluding the intervening utterance of the other speaker. The earlier utterance of the pair was treated as the prime, and the later utterance as the target. The remainder of the method was the same as in Experiments 1 and 2 (see Section 3.1).

5.2 Results

The results for the between-speaker and within-speaker adaptation are shown in Figure 10 and Figure 11 for same five phrase types as in the previous experiments.

A positive adaptation effect can be seen in the between-speaker data. For each phrase type, the adaptation probability is greater than the prior. In the within-speaker data, by comparison, the magnitude of the adaptation advantage is greatly decreased, in comparison with Figure 10. Indeed, for most phrase types, the adaptation probability is lower than the prior, i.e., we have a case of negative adaptation.

5.3 Discussion

The results of the two analyses confirm that adaptation can indeed be found between speakers in dialog, supporting the results of experimental work reviewed by Pickering and Garrod (2004). The results do not support the notion that priming is due to the facilitation of production processes within a given speaker, an account which would have predicted adaptation within speakers, but not between speakers.

The lack of clear positive adaptation effects in the within-speaker data is harder to explain—all current theories of priming would predict some effect here. One possibility is that such effects may have been obscured by decay processes: doing a within-speaker analysis entails skipping an intervening turn, in which priming effects were lost. We intend to address these concerns using more elaborate experimental designs in future work.

6 Conclusions

In this paper, we have demonstrated a robust, pervasive effect of parallelism for noun phrases. We found the tendency for structural repetition in two different corpora of written English, and also in a dialog corpus. The effect occurs in a wide range of contexts: within coordinate structures (Experiment 1), within sentences for NPs in an arbitrary structural configuration, between sentences, and within documents (Experiment 2). This strongly indicates that the parallelism effect is an instance of a general processing mechanism, such as syntactic priming (Bock, 1986), rather than specific to coordination, as suggested by (Frazier and Clifton, 2001). However, we also found that the parallelism effect is strongest in coordinate structures, which could explain why comprehension experiments so far failed to demonstrate the effect for other structural configurations (Frazier et al., 2000). We leave it to future work to explain *why* adaptation is much stronger in co-ordination: is co-ordination special because of extra constrains (i.e., some kind of expected contrast/comparison between co-ordinate sisters) or because of fewer constraints (i.e., both co-ordinate sisters have a similar grammatical role in the sentence)?

Another result (Experiment 3) is that the parallelism effect occurs between speakers in dialog. This finding is compatible with Pickering and Garrod's (2004) interactive alignment model, and strengthens the argument for parallelism as an instance of a general priming mechanism.

Previous experimental work has found parallelism effects, but only in comprehension data. The present work demonstrates that parallelism effects also occur in production data, which raises an interesting question of the relationship between the two data types. It has been hypothesized that the human language processing system is tuned to mirror the probability distributions in its environment, including the probabilities of syntactic structures (Mitchell et al., 1996). If this *tuning hypothesis* is correct, then the parallelism effect in comprehension data can be explained as an adaptation of the human parser to the prevalence of parallel structures in its environment (as approximated by corpus data) that we demonstrated in this paper.

Note that the results in this paper not only have an impact on theoretical issues regarding human sentence processing, but also on engineering problems in natural language processing, e.g., in probabilistic parsing. To avoid sparse data problems, probabilistic parsing models make strong independence assumptions; in particular, they generally assume that sentences are independent of each other. This is partly due to the fact it is difficult to parameterize the many possible dependencies which may occur between adjacent sentences. However, in this paper, we show that structure re-use is one possible way in which the independence assumption is broken. A simple and principled approach to handling structure re-use would be to use adaptation probabilities for probabilistic grammar rules, analogous to cache probabilities used in caching language models (Kuhn and de Mori, 1990). We are currently conducting further experiments to investigate of the effect of syntactic priming on probabilistic parsing.

References

Bock, J. Kathryn. 1986. Syntactic persistence in language production. *Cognitive Psychology* 18:355–387.

Bock, Kathryn and Helga Loebell. 1990. Framing sentences. *Cognition* 35(1):1–39.

Branigan, Holly P., Marin J. Pickering, and Janet F. McLean. 2005. Priming prepositional-phrase attachment during comprehension. *Journal of Experimental Psychology: Learning, Memory and Cognition* 31(3):468–481.

Carlson, Katy. 2002. The effects of parallelism and prosody on the processing of gapping structures. *Language and Speech* 44(1):1–26.

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.

Frazier, Lyn, Alan Munn, and Chuck Clifton. 2000. Processing coordinate structures. *Journal of Psycholinguistic Research* 29(4):343–370.

Frazier, Lyn, Lori Taft, Tom Roeper, Charles Clifton, and Kate Ehrlich. 1984. Parallel structure: A source of facilitation in sentence comprehension. *Memory and Cognition* 12(5):421–430.

Frazier, Lynn and Charles Clifton. 2001. Parsing coordinates and ellipsis: Copy α. *Syntax* 4(1):1–22.

Gries, Stefan T. 2005. Syntactic priming: A corpus-based approach. *Journal of Psycholinguistic Research* 35.

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.

Mauner, Gail, Michael K. Tanenhaus, and Greg Carlson. 1995. A note on parallelism effects in processing deep and surface verb-phrase anaphors. *Language and Cognitive Processes* 10:1–12.

Mitchell, Don C., Fernando Cuetos, Martin M. B. Corley, and Marc Brysbaert. 1996. Exposure-based models of human parsing: Evidence for the use of coarse-grained (non-lexical) statistical records. *Journal of Psycholinguistic Research* 24(6):469–488.

Pickering, Martin J. and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences* 27(2):169–225.

Szmrecsanyi, Benedikt. 2005. Creatures of habit: A corpus-linguistic analysis of persistence in spoken English. *Corpus Linguistics and Linguistic Theory* 1(1):113–149.

Using the Web as an Implicit Training Set: Application to Structural Ambiguity Resolution

Preslav Nakov and **Marti Hearst**
EECS and SIMS
University of California at Berkeley
Berkeley, CA 94720
nakov@cs.berkeley.edu, hearst@sims.berkeley.edu

Abstract

Recent work has shown that very large corpora can act as training data for NLP algorithms even without explicit labels. In this paper we show how the use of surface features and paraphrases in queries against search engines can be used to infer labels for structural ambiguity resolution tasks. Using unsupervised algorithms, we achieve 84% precision on PP-attachment and 80% on noun compound coordination.

1 Introduction

Resolution of structural ambiguity problems such as noun compound bracketing, prepositional phrase (PP) attachment, and noun phrase coordination requires using information about lexical items and their cooccurrences. This in turn leads to the data sparseness problem, since algorithms that rely on making decisions based on individual lexical items must have statistics about every word that may be encountered. Past approaches have dealt with the data sparseness problem by attempting to generalize from semantic classes, either manually built or automatically derived.

More recently, Banko and Brill (2001) have advocated for the creative use of very large text collections as an alternative to sophisticated algorithms and hand-built resources. They demonstrate the idea on a lexical disambiguation problem for which labeled examples are available "for free". The problem is to choose which of 2-3 commonly confused words (e.g., {*principle, principal*}) are appropriate for a given context. The labeled data comes "for free" by assuming that in most edited written text, the words are used correctly, so training can be done directly from the text. Banko and Brill (2001) show that even using a very simple algorithm, the results continue to improve log-linearly with more training data, even out to a billion words. A potential limitation of this approach is the question of how applicable it is for NLP problems more generally – how can we treat a large corpus as a labeled collection for a wide range of NLP tasks?

In a related strand of work, Lapata and Keller (2004) show that computing n-gram statistics over very large corpora yields results that are competitive with if not better than the best supervised and knowledge-based approaches on a wide range of NLP tasks. For example, they show that for the problem of noun compound bracketing, the performance of an n-gram based model computed using search engine statistics was not significantly different from the best supervised algorithm whose parameters were tuned and which used a taxonomy. They find however that these approaches generally fail to outperform supervised state-of-the-art models that are trained on smaller corpora, and so conclude that web-based n-gram statistics should be the baseline to beat.

We feel the potential of these ideas is not yet fully realized. We are interested in finding ways to further exploit the availability of enormous web corpora as implicit training data. This is especially important for structural ambiguity problems in which the decisions must be made on the basis of the behavior of individual lexical items. The trick is to figure out how to use information that is latent in the web as a corpus, and web search engines as query interfaces to that corpus.

In this paper we describe two techniques – *surface features* and *paraphrases* – that push the ideas of Banko and Brill (2001) and Lapata and Keller (2004) farther, enabling the use of statistics gathered from very large corpora in an unsupervised manner. In recent work (Nakov and Hearst, 2005) we showed that a variation of the techniques, when applied to the problem of noun compound bracketing, produces higher accuracy than Lapata and Keller (2004) and the best supervised results. In this paper we adapt the techniques to the structural disambiguation problems of prepositional phrase attachment and noun compound coordination.

2 Prepositional Phrase Attachment

A long-standing challenge for syntactic parsers is the attachment decision for prepositional phrases. In a configuration where a verb takes a noun complement that is followed by a PP, the problem arises of whether the PP attaches to the noun or to the verb. Consider the following contrastive pair of sentences:

(1) *Peter spent millions of dollars.* (noun)

(2) *Peter spent time with his family.* (verb)

In the first example, the PP *millions of dollars* attaches to the noun *millions*, while in the second the PP *with his family* attaches to the verb *spent*.

Past work on PP-attachment has often cast these associations as the quadruple (v, n_1, p, n_2), where v is the verb, n_1 is the head of the direct object, p is the preposition (the head of the PP) and n_2 is the head of the NP inside the PP. For example, the quadruple for (2) is (*spent, time, with, family*).

2.1 Related Work

Early work on PP-attachment ambiguity resolution relied on syntactic (e.g., "minimal attachment" and "right association") and pragmatic considerations. Most recent work can be divided into supervised and unsupervised approaches. Supervised approaches tend to make use of semantic classes or thesauri in order to deal with data sparseness problems. Brill and Resnik (1994) used the supervised transformation-based learning method and lexical and conceptual classes derived from WordNet, achieving 82% precision on 500 randomly selected examples. Ratnaparkhi et al. (1994) created a benchmark dataset of 27,937 quadruples (v, n_1, p, n_2), extracted from the Wall Street Journal. They found the human performance on this task to be 88%[1]. Using this dataset, they trained a maximum entropy model and a binary hierarchy of word classes derived by mutual information, achieving 81.6% precision. Collins and Brooks (1995) used a supervised back-off model to achieve 84.5% precision on the Ratnaparkhi test set. Stetina and Makoto (1997) use a supervised method with a decision tree and WordNet classes to achieve 88.1% precision on the same test set. Toutanova et al. (2004) use a supervised method that makes use of morphological and syntactic analysis and WordNet synsets, yielding 87.5% accuracy.

In the unsupervised approaches, the attachment decision depends largely on co-occurrence statistics drawn from text collections. The pioneering work in this area was that of Hindle and Rooth (1993). Using a partially parsed corpus, they calculate and compare lexical associations over subsets of the tuple (v, n_1, p), ignoring n_2, and achieve 80% precision at 80% recall.

More recently, Ratnaparkhi (1998) developed an unsupervised method that collects statistics from text annotated with part-of-speech tags and morphological base forms. An extraction heuristic is used to identify unambiguous attachment decisions, for example, the algorithm can assume a noun attachment if there is no verb within k words to the left of the preposition in a given sentence, among other conditions. This extraction heuristic uncovered 910K unique tuples of the form (v, p, n_2) and (n, p, n_2), although the results are very noisy, suggesting the correct attachment only about 69% of the time. The tuples are used as training data for classifiers, the best of which achieves 81.9% precision on the Ratnaparkhi test set. Pantel and Lin (2000) describe an unsupervised method that uses a collocation database, a thesaurus, a dependency parser, and a large corpus (125M words), achieving 84.3% precision on the Ratnaparkhi test set. Using sim-

[1]When presented with a whole sentence, average humans score 93%.

ple combinations of web-based n-grams, Lapata and Keller (2005) achieve lower results, in the low 70's.

Using a different collection consisting of German PP-attachment decisions, Volk (2000) uses the web to obtain n-gram counts. He compared $\Pr(p|n_1)$ to $\Pr(p|v)$, where $\Pr(p|x) = \#(x,p)/\#(x)$. Here x can be n_1 or v. The bigram frequencies $\#(x,p)$ were obtained using the Altavista NEAR operator. The method was able to make a decision on 58% of the examples with a precision of 75% (baseline 63%). Volk (2001) then improved on these results by comparing $\Pr(p,n_2|n_1)$ to $\Pr(p,n_2|v)$. Using inflected forms, he achieved P=75% and R=85%.

Calvo and Gelbukh (2003) experimented with a variation of this, using exact phrases instead of the NEAR operator. For example, to disambiguate *Veo al gato con un telescopio*, they compared frequencies for phrases such as "ver con telescopio" and "gato con telescopio". They tested this idea on 181 randomly chosen Spanish disambiguation examples, labelling 89.5% recall with a precision of 91.97%.

2.2 Models and Features

2.2.1 n-gram Models

We computed two co-occurrence models;

(*i*) $\Pr(p|n_1)$ vs. $\Pr(p|v)$

(*ii*) $\Pr(p,n_2|n_1)$ vs. $\Pr(p,n_2|v)$.

Each of these was computed two different ways: using $\Pr$ (probabilities) and $\#$ (frequencies). We estimate the n-gram counts using exact phrase queries (with inflections, derived from WordNet 2.0) using the MSN Search Engine. We also allow for determiners, where appropriate, e.g., between the preposition and the noun when querying for $\#(p,n_2)$. We add up the frequencies for all possible variations. Web frequencies were reliable enough and did not need smoothing for (*i*), but for (*ii*), smoothing using the technique described in Hindle and Rooth (1993) led to better recall. We also tried back-off from (*ii*) to (*i*), as well as back-off plus smoothing, but did not find improvements over smoothing alone. We found n-gram counts to be unreliable when pronouns appear in the test set rather than nouns, and disabled them in these cases. Such examples can still be handled by paraphrases or surface features (see below).

2.2.2 Web-Derived Surface Features

Authors sometimes (consciously or not) disambiguate the words they write by using surface-level markers to suggest the correct meaning. We have found that exploiting these markers, when they occur, can prove to be very helpful for making disambiguation decisions. The enormous size of web search engine indexes facilitates finding such markers frequently enough to make them useful.

For example, *John opened the door with a key* is a difficult verb attachment example because doors, keys, and opening are all semantically related. To determine if this should be a verb or a noun attachment, we search for cues that indicate which of these terms tend to associate most closely. If we see parentheses used as follows:

"open the door (with a key)"

this suggests a verb attachment, since the parentheses signal that "with a key" acts as its own unit. Similarly, hyphens, colons, capitalization, and other punctuation can help signal disambiguation decisions. For *Jean ate spaghetti with sauce*, if we see

"eat: spaghetti with sauce"

this suggests a noun attachment.

Table 1 illustrates a wide variety of surface features, along with the attachment decisions they are assumed to suggest (events of frequency 1 have been ignored). The surface features for PP-attachment have low recall: most of the examples have no surface features extracted.

We gather the statistics needed by issuing queries to web search engines. Unfortunately, search engines usually ignore punctuation characters, thus preventing querying directly for terms containing hyphens, brackets, etc. We collect these numbers indirectly by issuing queries with exact phrases and then post-processing the top 1,000 resulting summaries[2], looking for the surface features of interest. We use Google for both the surface feature and paraphrase extractions (described below).

2.2.3 Paraphrases

The second way we extend the use of web counts is by paraphrasing the relation of interest and seeing if it can be found in its alternative form, which

[2]We often obtain more than 1,000 summaries per example because we usually issue multiple queries per surface pattern, by varying inflections and inclusion of determiners.

suggests the correct attachment decision. We use the following patterns along with their associated attachment predictions:

(1)	$v\ n_2\ n_1$	(noun)
(2)	$v\ p\ n_2\ n_1$	(verb)
(3)	$p\ n_2$ * $v\ n_1$	(verb)
(4)	$n_1\ p\ n_2\ v$	(noun)
(5)	v *pronoun* $p\ n_2$	(verb)
(6)	*be* $n_1\ p\ n_2$	(noun)

The idea behind Pattern (1) is to determine if "$n_1\ p\ n_2$" can be expressed as a noun compound; if this happens sufficiently often, we can predict a noun attachment. For example, *meet/*v *demands/*n_1 *from/*p *customers/*n_2 becomes *meet/*v *the customers/*n_2 *demands/*n_1.

Note that the pattern could wrongly target ditransitive verbs: e.g., it could turn *gave/*v *an apple/*n_1 *to/*p *him/*n_2 into *gave/*v *him/*n_2 *an apple/*n_1. To prevent this, we do not allow a determiner before n_1, but we do require one before n_2. In addition, we disallow the pattern if the preposition is *to* and we require both n_1 and n_2 to be nouns (as opposed to numbers, percents, pronouns, determiners etc.).

Pattern (2) predicts a verb attachment. It presupposes that "$p\ n_2$" is an indirect object of the verb v and tries to switch it with the direct object n_1, e.g., *had/*v *a program/*n_1 *in/*p *place/*n_2 would be transformed into *had/*v *in/*p *place/*n_2 *a program/*n_1. We require n_1 to be preceded by a determiner (to prevent "$n_2\ n_1$" forming a noun compound).

Pattern (3) looks for appositions, where the PP has moved in front of the verb, e.g., *to/*p *him/*n_2 *I gave/*v *an apple/*n_1. The symbol * indicates a wildcard position where we allow up to three intervening words.

Pattern (4) looks for appositions, where the PP has moved in front of the verb together with n_1. It would transform *shaken/*v *confidence/*n_1 *in/*p *markets/*n_2 into *confidence/*n_1 *in/*p *markets/*n_2 *shaken/*v.

Pattern (5) is motivated by the observation that if n_1 is a pronoun, this suggests a verb attachment (Hindle and Rooth, 1993). (A separate feature checks if n_1 is a pronoun.) The pattern substitutes n_1 with a dative pronoun (we allow *him* and *her*), e.g., it will convert *put/*v *a client/*n_1 *at/*p *odds/*n_2 into *put/*v *him at/*p *odds/*n_2.

Pattern (6) is motivated by the observation that the verb *to be* is typically used with a noun attachment. (A separate feature checks if v is a form of the verb *to be*.) The pattern substitutes v with *is* and *are*, e.g. it will turn *eat/*v *spaghetti/*n_1 *with/*p *sauce/*n_2 into *is spaghetti/*n_1 *with/*p *sauce/*n_2.

These patterns all allow for determiners where appropriate, unless explicitly stated otherwise. For a given example, a prediction is made if at least one instance of the pattern has been found.

2.3 Evaluation

For the evaluation, we used the test part (3,097 examples) of the benchmark dataset by Ratnaparkhi et al. (1994). We used all 3,097 test examples in order to make our results directly comparable.

Unfortunately, there are numerous errors in the test set[3]. There are 149 examples in which a bare determiner is labeled as n_1 or n_2 rather than the actual head noun. Supervised algorithms can compensate for this problem by learning from the training set that "the" can act as a noun in this collection, but unsupervised algorithms cannot.

In addition, there are also around 230 examples in which the nouns contain special symbols like: %, slash, &, ', which are lost when querying against a search engine. This poses a problem for our algorithm but is not a problem with the test set itself.

The results are shown in Table 2. Following Ratnaparkhi (1998), we predict a noun attachment if the preposition is *of* (a very reliable heuristic). The table shows the performance for each feature in isolation (excluding examples whose preposition is *of*). The surface features are represented by a single score in Table 2: for a given example, we sum up separately the number of noun- and verb-attachment pattern matches, and assign the attachment with the larger number of matches.

We combine the bold rows of Table 2 in a majority vote (assigning noun attachment to all *of* instances), obtaining P=85.01%, R=91.77%. To get 100% recall, we assign all undecided cases to *verb* (since the majority of the remaining non-*of* instances attach to the verb, yielding P=83.63%, R=100%. We show 0.95-level confidence intervals for the precision, computed by a general method based on constant chi-square boundaries (Fleiss, 1981).

A test for statistical significance reveals that our results are as strong as those of the leading unsuper-

[3]Ratnaparkhi (1998) notes that the test set contains errors, but does not correct them.

Example	Predicts	P(%)	R(%)
open Door with a key	noun	100.00	0.13
(open) door with a key	noun	66.67	0.28
open (door with a key)	noun	71.43	0.97
open - door with a key	noun	69.70	1.52
open / door with a key	noun	60.00	0.46
open, door with a key	noun	65.77	5.11
open: door with a key	noun	64.71	1.57
open; door with a key	noun	60.00	0.23
open. door with a key	noun	64.13	4.24
open? door with a key	noun	83.33	0.55
open! door with a key	noun	66.67	0.14
open door With a Key	verb	0.00	0.00
(open door) with a key	verb	50.00	0.09
open door (with a key)	verb	73.58	2.44
open door - with a key	verb	68.18	2.03
open door / with a key	verb	100.00	0.14
open door, with a key	verb	58.44	7.09
open door: with a key	verb	70.59	0.78
open door; with a key	verb	75.00	0.18
open door. with a key	verb	60.77	5.99
open door! with a key	verb	100.00	0.18

Table 1: **PP-attachment surface features.** Precision and recall shown are across all examples, not just the door example shown.

Model	**P**(%)	**R**(%)
Baseline (noun attach)	41.82	100.00
$\#(x,p)$	58.91	83.97
$\Pr(p\|x)$	66.81	83.97
$\Pr(p\|x)$ smoothed	**66.81**	83.97
$\#(x,p,n_2)$	65.78	81.02
$\Pr(p,n_2\|x)$	68.34	81.62
$\Pr(p,n_2\|x)$ smoothed	**68.46**	83.97
(1) "$v\ n_2\ n_1$"	**59.29**	22.06
(2) "$p\ n_2\ v\ n_1$"	**57.79**	71.58
(3) "n_1 * $p\ n_2\ v$"	**65.78**	20.73
(4) "$v\ p\ n_2\ n_1$"	**81.05**	8.75
(5) "v *pronoun* $p\ n_2$"	**75.30**	30.40
(6) "*be* $n_1\ p\ n_2$"	**63.65**	30.54
n_1 is *pronoun*	**98.48**	3.04
v is *to be*	**79.23**	9.53
Surface features (summed)	**73.13**	9.26
Maj. vote, of → noun	85.01±1.21	91.77
Maj. vote, of → noun, N/A → verb	***83.63±1.30***	***100.00***

Table 2: **PP-attachment results, in percentages.**

vised approach on this collection (Pantel and Lin, 2000). Unlike that work, we do not require a collocation database, a thesaurus, a dependency parser, nor a large domain-dependent text corpus, which makes our approach easier to implement and to extend to other languages.

3 Coordination

Coordinating conjunctions (*and, or, but*, etc.) pose major challenges to parsers and their proper handling is essential for the understanding of the sentence. Consider the following "cooked" example:

The Department of Chronic Diseases ***and*** *Health Promotion leads* ***and*** *strengthens global efforts to prevent* ***and*** *control chronic diseases* ***or*** *disabilities* ***and*** *to promote health* ***and*** *quality of life.*

Conjunctions can link two words, two constituents (e.g., NPs), two clauses or even two sentences. Thus, the first challenge is to identify the boundaries of the conjuncts of each coordination. The next problem comes from the interaction of the coordinations with other constituents that attach to its conjuncts (most often prepositional phrases). In the example above we need to decide between *[health and [quality of life]]* and *[[health and quality] of life]*. From a semantic point of view, we need to determine whether the *or* in *chronic diseases or disabilities* really means *or* or is used as an *and* (Agarwal and Boggess, 1992). Finally, we need to choose between a *non-elided* and an *elided* reading: *[[chronic diseases] or disabilities]* vs. *[chronic [diseases or disabilities]]*.

Below we focus on a special case of the latter problem: noun compound (NC) coordination. Consider the NC *car and truck production*. Its real meaning is *car production and truck production*. However, due to the principle of economy of expression, the first instance of *production* has been compressed out by means of ellipsis. By contrast, in *president and chief executive*, *president* is simply linked to *chief executive*. There is also an all-way coordination, where the conjunct is part of the whole, as in *Securities and Exchange Commission*.

More formally, we consider configurations of the kind $n_1\ c\ n_2\ h$, where n_1 and n_2 are nouns, c is a coordination (*and* or *or*) and h is the head noun[4]. The task is to decide whether there is an ellipsis or not, independently of the local context. Syntactically, this can be expressed by the following bracketings: $[[n_1\ c\ n_2]\ h]$ versus $[n_1\ c\ [n_2\ h]]$. (Collins' parser (Collins, 1997) always predicts a flat NP for such configurations.) In order to make the task more

[4]The configurations of the kind $n\ h_1\ c\ h_2$ (e.g., *company/n cars/*h_1 *and/c trucks/*h_2) can be handled in a similar way.

realistic (from a parser's perspective), we ignore the option of all-way coordination and try to predict the bracketing in Penn Treebank (Marcus et al., 1994) for configurations of this kind. The Penn Treebank brackets NCs with ellipsis as, e.g.,
(NP car/NN and/CC truck/NN production/NN).
and without ellipsis as
(NP (NP president/NN) and/CC (NP chief/NN executive/NN))
The NPs with ellipsis are flat, while the others contain internal NPs. The all-way coordinations can appear bracketed either way and make the task harder.

3.1 Related Work

Coordination ambiguity is under-explored, despite being one of the three major sources of structural ambiguity (together with prepositional phrase attachment and noun compound bracketing), and belonging to the class of ambiguities for which the number of analyses is the number of binary trees over the corresponding nodes (Church and Patil, 1982), and despite the fact that conjunctions are among the most frequent words.

Rus et al. (2002) present a deterministic rule-based approach for bracketing *in context* of coordinated NCs of the kind n_1 c n_2 h, as a necessary step towards logical form derivation. Their algorithm uses POS tagging, syntactic parses, semantic senses of the nouns (manually annotated), lookups in a semantic network (WordNet) and the type of the coordination conjunction to make a 3-way classification: ellipsis, no ellipsis and all-way coordination. Using a back-off sequence of 3 different heuristics, they achieve 83.52% precision (baseline 61.52%) on a set of 298 examples. When 3 additional context-dependent heuristics and 224 additional examples with local contexts are added, the precision jumps to 87.42% (baseline 52.35%), with 71.05% recall.

Resnik (1999) disambiguates two kinds of patterns: n_1 and n_2 n_3 and n_1 n_2 and n_3 n_4 (e.g., *[food/*n_1 *[handling/*n_2 *and/*c *storage/*n_3*] procedures/*n_4*]*). While there are two options for the former (all-way coordinations are not allowed), there are 5 valid bracketings for the latter. Following Kurohashi and Nagao (1992), Resnik makes decisions based on similarity of form (i.e., number agreement: P=53%, R=90.6%), similarity of meaning (P=66%, R=71.2%) and conceptual association (P=75.0%, R=69.3%). Using a decision tree to combine the three information sources, he achieves 80% precision (baseline 66%) at 100% recall for the 3-noun coordinations. For the 4-noun coordinations the precision is 81.6% (baseline 44.9%), 85.4% recall.

Example	Predicts	P(%)	R(%)
(buy) and sell orders	NO ellipsis	33.33	1.40
buy (and sell orders)	NO ellipsis	70.00	4.67
buy: and sell orders	NO ellipsis	0.00	0.00
buy; and sell orders	NO ellipsis	66.67	2.80
buy. and sell orders	NO ellipsis	68.57	8.18
buy[...] and sell orders	NO ellipsis	49.00	46.73
buy- and sell orders	ellipsis	77.27	5.14
buy and sell / orders	ellipsis	50.54	21.73
(buy and sell) orders	ellipsis	92.31	3.04
buy and sell (orders)	ellipsis	90.91	2.57
buy and sell, orders	ellipsis	92.86	13.08
buy and sell: orders	ellipsis	93.75	3.74
buy and sell; orders	ellipsis	100.00	1.87
buy and sell. orders	ellipsis	93.33	7.01
buy and sell[...] orders	ellipsis	85.19	18.93

Table 3: **Coordination surface features.** Precision and recall shown are across all examples, not just the *buy and sell orders* shown.

Chantree et al. (2005) cover a large set of ambiguities, not limited to nouns. They allow the head word to be a noun, a verb or an adjective, and the modifier to be an adjective, a preposition, an adverb, etc. They extract distributional information from the British National Corpus and distributional similarities between words, similarly to (Resnik, 1999). In two different experiments they achieve P=88.2%, R=38.5% and P=80.8%, R=53.8% (baseline P=75%).

Goldberg (1999) resolves the *attachment of ambiguous coordinate phrases* of the kind n_1 p n_2 c n_3, e.g., *box/*n_1 *of/*p *chocolates/*n_2 *and/*c *roses/*n_3. Using an adaptation of the algorithm proposed by Ratnaparkhi (1998) for PP-attachment, she achieves P=72% (baseline P=64%), R=100.00%.

Agarwal and Boggess (1992) focus on the *identification of the conjuncts of coordinate conjunctions*. Using POS and case labels in a deterministic algorithm, they achieve P=81.6%. Kurohashi and Nagao (1992) work on the same problem for Japanese. Their algorithm looks for similar word sequences among with sentence simplification, and achieves a precision of 81.3%.

3.2 Models and Features

3.2.1 n-gram Models

We use the following n-gram models:

(*i*) $\#(n_1, h)$ vs. $\#(n_2, h)$

(*ii*) $\#(n_1, h)$ vs. $\#(n_1, c, n_2)$

Model (*i*) compares how likely it is that n_1 modifies h, as opposed to n_2 modifying h. Model (*ii*) checks which association is stronger: between n_1 and h, or between n_1 and n_2. Regardless of whether the coordination is *or* or *and*, we query for both and we add up the corresponding counts.

3.2.2 Web-Derived Surface Features

The set of surface features is similar to the one we used for PP-attachment. These are brackets, slash, comma, colon, semicolon, dot, question mark, exclamation mark, and any character. There are two additional ellipsis-predicting features: a dash after n_1 and a slash after n_2, see Table 3.

3.2.3 Paraphrases

We use the following paraphrase patterns:

(1)	n_2 *c* n_1 h	(ellipsis)
(2)	n_2 h *c* n_1	(NO ellipsis)
(3)	n_1 h *c* n_2 h	(ellipsis)
(4)	n_2 h *c* n_1 h	(ellipsis)

If matched frequently enough, each of these patterns predicts the coordination decision indicated in parentheses. If found only infrequently or not found at all, the opposite decision is made. Pattern (1) switches the places of n_1 and n_2 in the coordinated NC. For example, *bar and pie graph* can easily become *pie and bar graph*, which favors ellipsis. Pattern (2) moves n_2 and h together to the left of the coordination conjunction, and places n_1 to the right. If this happens frequently enough, there is no ellipsis. Pattern (3) inserts the elided head h after n_1 with the hope that if there is ellipsis, we will find the full phrase elsewhere in the data. Pattern (4) combines pattern (1) and pattern (3); it not only inserts h after n_1 but also switches the places of n_1 and n_2.

As shown in Table 4, we included four of the heuristics by Rus et al. (2002). Heuristic 1 predicts no coordination when n_1 and n_2 are the same, e.g., *milk and milk products*. Heuristics 2 and 3 perform a lookup in WordNet and we did not use them. Heuristics 4, 5 and 6 exploit the local context, namely the adjectives modifying n_1 and/or n_2. Heuristic 4 predicts no ellipsis if both n_1 and n_2 are modified by adjectives. Heuristic 5 predicts ellipsis if the coordination is *or* and n_1 is modified by an adjective, but n_2 is not. Heuristic 6 predicts no ellipsis if n_1 is not modified by an adjective, but n_2 is. We used versions of heuristics 4, 5 and 6 that check for determiners rather than adjectives.

Model	**P**(%)	**R**(%)
Baseline: ellipsis	56.54	100.00
(n_1, h) vs. (n_2, h)	**80.33**	28.50
(n_1, h) vs. (n_1, c, n_2)	61.14	45.09
(n_2, c, n_1, h)	**88.33**	14.02
(n_2, h, c, n_1)	**76.60**	21.96
(n_1, h, c, n_2, h)	**75.00**	6.54
(n_2, h, c, n_1, h)	**78.67**	17.52
Heuristic 1	**75.00**	0.93
Heuristic 4	64.29	6.54
Heuristic 5	61.54	12.15
Heuristic 6	**87.09**	7.24
Number agreement	**72.22**	46.26
Surface sum	**82.80**	21.73
Majority vote	83.82	80.84
Majority vote, N/A → no ellipsis	***80.61***	***100.00***

Table 4: **Coordination results, in percentages.**

Finally, we included the number agreement feature (Resnik, 1993): (a) if n_1 and n_2 match in number, but n_1 and h do not, predict ellipsis; (b) if n_1 and n_2 do not match in number, but n_1 and h do, predict no ellipsis; (c) otherwise leave undecided.

3.3 Evaluation

We evaluated the algorithms on a collection of 428 examples extracted from the Penn Treebank. On extraction, determiners and non-noun modifiers were allowed, but the program was only presented with the quadruple (n_1, c, n_2, h). As Table 4 shows, our overall performance of 80.61 is on par with other approaches, whose best scores fall into the low 80's for precision. (Direct comparison is not possible, as the tasks and datasets all differ.)

As Table 4 shows, n-gram model (*i*) performs well, but n-gram model (*ii*) performs poorly, probably because the (n_1, c, n_2) contains three words, as opposed to two for the alternative (n_1, h), and thus a priori is less likely to be observed.

The surface features are less effective for resolving coordinations. As Table 3 shows, they are very good predictors of ellipsis, but are less reliable when predicting NO ellipsis. We combine the bold rows of Table 4 in a majority vote, obtaining P=83.82%, R=80.84%. We assign all undecided cases to no ellipsis, yielding P=80.61%, R=100%.

4 Conclusions and Future Work

We have shown that simple unsupervised algorithms that make use of bigrams, surface features and paraphrases extracted from a very large corpus are effective for several structural ambiguity resolutions tasks, yielding results competitive with the best unsupervised results, and close to supervised results. The method does not require labeled training data, nor lexicons nor ontologies. We think this is a promising direction for a wide range of NLP tasks. In future work we intend to explore better-motivated evidence combination algorithms and to apply the approach to other NLP problems.

Acknowledgements. This research was supported by NSF DBI-0317510 and a gift from Genentech.

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of ACL*.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of ACL*.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of COLING*.

Hiram Calvo and Alexander Gelbukh. 2003. Improving prepositional phrase attachment disambiguation using the web as corpus. In *Progress in Pattern Recognition, Speech and Image Analysis: 8th Iberoamerican Congress on Pattern Recognition, CIARP 2003*.

Francis Chantree, Adam Kilgarriff, Anne De Roeck, and Alistair Willis. 2005. Using a distributional thesaurus to resolve coordination ambiguities. In *Technical Report 2005/02*. The Open University, UK.

Kenneth Church and Ramesh Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Amer. J. of Computational Linguistics*, 8(3-4):139–149.

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of EMNLP*, pages 27–38.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, pages 16–23.

Joseph Fleiss. 1981. *Statistical Methods for Rates and Proportions (2nd Ed.)*. John Wiley & Sons, New York.

Miriam Goldberg. 1999. An unsupervised model for statistically determining coordinate phrase attachment. In *Proceedings of ACL*.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Sadao Kurohashi and Makoto Nagao. 1992. Dynamic programming method for analyzing conjunctive structures in japanese. In *Proceedings of COLING*, volume 1.

Mirella Lapata and Frank Keller. 2004. The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. In *Proceedings of HLT-NAACL*, pages 121–128, Boston.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2:1–31.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL 2005*.

Patrick Pantel and Dekang Lin. 2000. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *Proceedings of ACL*.

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.*, pages 250–255.

Adwait Ratnaparkhi. 1998. Statistical models for unsupervised prepositional phrase attachment. In *Proceedings of COLING-ACL*, volume 2, pages 1079–1085.

Philip Resnik. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania, UMI Order No. GAX94-13894.

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11:95–130.

Vasile Rus, Dan Moldovan, and Orest Bolohan. 2002. Bracketing compound nouns for logic form derivation. In Susan M. Haller and Gene Simmons, editors, *FLAIRS Conference*. AAAI Press.

Jiri Stetina and Makoto. 1997. Corpus based PP attachment ambiguity resolution with a semantic dictionary. In *Proceedings of WVLC*, pages 66–80.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *Proceedings of ICML*.

Martin Volk. 2000. Scaling up. using the WWW to resolve PP attachment ambiguities. In *Proceedings of Konvens-2000. Sprachkommunikation*.

Martin Volk. 2001. Exploiting the WWW as a corpus to resolve PP attachment ambiguities. In *Proc. of Corpus Linguistics*.

Paradigmatic Modifiability Statistics for the Extraction of Complex Multi-Word Terms

Joachim Wermter
Jena University
Language & Information Engineering
JULIE Lab
Friedrich-Schiller-Universität Jena
Fürstengraben 30
D-07743 Jena, Germany
joachim.wermter@uni-jena.de

Udo Hahn
Jena University
Language & Information Engineering
JULIE Lab
Friedrich-Schiller-Universität Jena
Fürstengraben 30
D-07743 Jena, Germany
udo.hahn@uni-jena.de

Abstract

We here propose a new method which sets apart domain-specific terminology from common non-specific noun phrases. It is based on the observation that terminological multi-word groups reveal a considerably lesser degree of distributional variation than non-specific noun phrases. We define a measure for the observable amount of paradigmatic modifiability of terms and, subsequently, test it on bigram, trigram and quadgram noun phrases extracted from a 104-million-word biomedical text corpus. Using a community-wide curated biomedical terminology system as an evaluation gold standard, we show that our algorithm significantly outperforms a variety of standard term identification measures. We also provide empirical evidence that our methodolgy is essentially domain- and corpus-size-independent.

1 Introduction

As we witness the ever-increasing proliferation of volumes of medical and biological documents, the available dictionaries and terminological systems cannot keep up with this pace of growth and, hence, become more and more incomplete. What's worse, the constant stream of new terms is increasingly getting unmanageable because human curators are in the loop. The costly, often error-prone and time-consuming nature of manually identifying new terminology from the most recent literature calls for advanced procedures which can automatically assist database curators in the task of assembling, updating and maintaining domain-specific controlled vocabularies. Whereas the recognition of single-word terms usually does not pose any particular challenges, the vast majority of biomedical or any other domain-specific terms typically consists of multi-word units.[1] Unfortunately these are much more difficult to recognize and extract than their singleton counterparts. Moreover, although the need to assemble and extend technical and scientific terminologies is currently most pressing in the biomedical domain, virtually any (sub-)field of human research/expertise in which we deal with terminologically structured knowledge calls for high-performance terminology identification and extraction methods. We want to target exactly this challenge.

2 Related Work

The automatic extraction of complex multi-word terms from domain-specific corpora is already an active field of research (cf., e.g., for the biomedical domain Rindflesch et al. (1999), Collier et al. (2002), Bodenreider et al. (2002), or Nenadić et al. (2003)). Typically, in all of these approaches term candidates are collected from texts by various forms of linguistic filtering (part-of-speech tagging, phrase chunking, etc.), through which candidates obeying various linguistic patterns are identified (e.g., *noun-noun*, *adjective-noun-noun* combinations). These candidates are then submitted to frequency- or statistically-based evidence measures

[1]Nakagawa and Mori (2002) claim that more than 85% of domain-specific terms are multi-word units.

(such as the C-value (Frantzi et al., 2000)), which compute scores indicating to what degree a candidate qualifies as a term. Term mining, as a whole, is a complex process involving several other components (orthographic and morphological normalization, acronym detection, conflation of term variants, term context, term clustering; cf. Nenadić et al. (2003)). Still, the measure which assigns a *termhood value* to a term candidate is the essential building block of any term identification system.

For multi-word automatic term recognition (ATR), the C-value approach (Frantzi et al., 2000; Nenadić et al., 2004), which aims at improving the extraction of nested terms, has been one of the most widely used techniques in recent years. Other potential association measures are mutual information (Damerau, 1993) and the whole battery of statistical and information-theoretic measures (t-test, log-likelihood, entropy) which are typically employed for the extraction of general-language collocations (Manning and Schütze, 1999; Evert and Krenn, 2001). While these measures have their statistical merits in terminology identification, it is interesting to note that they only make little use of linguistic properties inherent to complex terms.[2]

More linguistically oriented work on ATR by Daille (1996) or on term variation by Jacquemin (1999) builds on the deep syntactic analysis of term candidates. This includes morphological and head-modifier dependency analysis and thus presupposes accurate, high-quality parsing which, for sublanguages at least, can only be achieved by a highly domain-dependent type of grammar. As sublanguages from different domains usually reveal a high degree of syntactic variability among each other (e.g., in terms of POS distribution, syntactic patterns), this property makes it difficult to port grammatical specifications to different domains.

Therefore, one may wonder whether there are cross-domain linguistic properties which might be beneficial to ATR and still could be accounted for by only *shallow* syntactic analysis. In this paper, we propose the *limited paradigmatic modifiability* of terms as a criterion which meets these requirements and will elaborate on it in detail in Subsection 3.3.

[2]A notable exception is the C-value method which incorporates a term's likelihood of being nested in other multi-word units.

3 Methods and Experiments

3.1 Text Corpus

We collected a biomedical training corpus of approximately 513,000 MEDLINE abstracts using the following query composed of MESH terms from the biomedical domain: *transcription factors*, *blood cells* and *human*.[3] We then annotated the resulting 104-million-word corpus with the GENIA part-of-speech tagger[4] and identified noun phrases (NPs) with the YAMCHA chunker (Kudo and Matsumoto, 2001). We restrict our study to NP recognition (i.e., determining the extension of a noun phrase but refraining from assigning any internal constituent structure to that phrase), because the vast majority of technical or scientific terms surface as noun phrases (Justeson and Katz, 1995). We filtered out a number of stop words (determiners, pronouns, measure symbols, etc.) and also ignored noun phrases with coordination markers (*"and"*, *"or"*, etc.).[5]

n-gram length	cut-off	NP term candidates tokens	types
bigrams	no cut-off	5,920,018	1,055,820
	$c \geq 10$	4,185,427	67,308
trigrams	no cut-off	3,110,786	1,655,440
	$c \geq 8$	1,053,651	31,017
quadgrams	no cut-off	1,686,745	1,356,547
	$c \geq 6$	222,255	10,838

Table 1: Frequency distribution for noun phrase term candidate tokens and types for the MEDLINE text corpus

In order to obtain the term candidate sets (see Table 1), we counted the frequency of occurrence of noun phrases in our training corpus and categorized them according to their length. For this study, we restricted ourselves to noun phrases of length 2 (word bigrams), length 3 (word trigrams) and length 4 (word quadgrams). Morphological normalization of term candidates has shown to be beneficial for ATR (Nenadić et al., 2004). We thus normalized the nom-

[3]MEDLINE (`http://www.ncbi.nlm.nih.gov`) is the largest biomedical bibliographic database. For information retrieval purposes, all of its abstracts are indexed with a controlled indexing vocabulary, the *Medical Subject Headings* (MESH, 2004).

[4]`http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/postagger/`

[5]Of course, terms can also be contained within coordinative structures (e.g., *"B and T cell"*). However, analyzing their inherent ambiguity is a complex syntactic operation, with a comparatively marginal benefit for ATR (Nenadić et al., 2004).

inal head of each noun phrase (typically the rightmost noun in English) via the full-form UMLS SPECIALIST LEXICON (UMLS, 2004), a large repository of both general-language and domain-specific (medical) vocabulary. To eliminate noisy low-frequency data (cf. also Evert and Krenn (2001)), we defined different frequency cut-off thresholds, c, for the bigram, trigram and quadgram candidate sets and only considered candidates above these thresholds.

3.2 Evaluating Term Extraction Quality

Typically, terminology extraction studies evaluate the goodness of their algorithms by having their ranked output examined by domain experts who identify the true positives among the ranked candidates. There are several problems with such an approach. First, very often only one such expert is consulted and, hence, inter-annotator agreement cannot be determined (as, e.g., in the studies of Frantzi et al. (2000) or Collier et al. (2002)). Furthermore, what constitutes a relevant term for a particular domain may be rather difficult to decide – even for domain experts – when judges are just exposed to a list of candidates without any further context information. Thus, rather than relying on *ad hoc* human judgment in identifying true positives in a candidate set, as an alternative we may take already existing terminolgical resources into account. They have evolved over many years and usually reflect community-wide consensus achieved by expert committees. With these considerations in mind, the biomedical domain is an ideal test bed for evaluating the goodness of ATR algorithms because it hosts one of the most extensive and most carefully curated terminological resources, viz. the UMLS METATHESAURUS (UMLS, 2004). We will then take the mere existence of a term in the UMLS as the decision criterion whether or not a candidate term is also recognized as a biomedical term.

Accordingly, for the purpose of evaluating the quality of different measures in recognizing multi-word terms from the biomedical literature, we assign every word bigram, trigram, and quadgram in our candidate sets (see Table 1) the status of being a term (i.e., a true positive), if it is found in the 2004 edition of the UMLS METATHESAURUS.[6] For example, the word trigram *"long terminal repeat"* is listed as a term in one of the UMLS vocabularies, *viz.* MESH (2004), whereas *"t cell response"* is not. Thus, among the 67,308 word bigram candidate types, 14,650 (21.8%) were identified as true terms; among the 31,017 word trigram candidate types, their number amounts to 3,590 (11.6%), while among the 10,838 word quadgram types, 873 (8.1%) were identified as true terms.[7]

[6]We exclude UMLS vocabularies not relevant for molecular biology, such as nursing and health care billing codes.

3.3 Paradigmatic Modifiability of Terms

For most standard association measures utilized for terminology extraction, the frequency of occurrence of the term candidates either plays a major role (e.g., C-value), or has at least a significant impact on the assignment of the degree of termhood (e.g., t-test). However, frequency of occurrence in a training corpus may be misleading regarding the decision whether or not a multi-word expression is a term. For example, taking the two trigram multiword expressions from the previous subsection, the non-term "*t cell response*" appears 2410 times in our 104-million-word MEDLINE corpus, whereas the term "*long terminal repeat*" (long repeating sequences of DNA) only appears 434 times (see also Tables 2 and 3 below).

The linguistic property around which we built our measure of termhood is the *limited paradigmatic modifiability* of multi-word terminological units. A multi-word expression such as "*long terminal repeat*" contains three token slots in which slot 1 is filled by "*long*", slot 2 by "*terminal*" and slot 3 by "*repeat*". The *limited paradigmatic modifiability* of such a trigram is now defined by the probability with which one or more such slots *cannot* be filled by other tokens. We estimate the likelihood of precluding the appearance of alternative tokens in particular slot positions by employing the standard combinatory formula without repetitions. For an n-gram (of size n) to select k slots (i.e., in an unordered selection) we thus define:

$$C(n,k) = \frac{n!}{k!(n-k)!} \tag{1}$$

[7]As can be seen, not only does the number of candidate types drop with increasing n-gram length but also the proportion of true terms. In fact, their proportion drops more sharply than can actually be seen from the above data because the various cut-off thresholds have a leveling effect.

For example, for $n = 3$ (word trigram) and $k = 1$ and $k = 2$ slots, there are three possible selections for each k for "*long terminal repeat*" and for "*t cell response*" (see Tables 2 and 3). k is actually a placeholder for any possible token (and its frequency) which fills this position in the training corpus.

n-gram		freq	$P\text{-}Mod$ (k=1,2)
long terminal repeat		434	0.03
k slots	possible selections *sel*	freq	mod_{sel}
$k = 1$	k_1 terminal repeat	460	0.940
	long k_2 repeat	448	0.970
	long terminal k_3	436	0.995
			mod_1 =0.91
$k = 2$	k_1 k_2 repeat	1831	0.23
	k_1 terminal k_3	1062	0.41
	long k_2 k_3	1371	0.32
			mod_2 =0.03

Table 2: $P\text{-}Mod$ and k-modifiabilities for $k = 1$ and $k = 2$ for the trigram term *"long terminal repeat"*

n-gram		freq	$P\text{-}Mod$ (k=1,2)
t cell response		2410	0.00005
k slots	possible selections *sel*	freq	mod_{sel}
$k = 1$	k_1 cell response	3248	0.74
	t k_2 response	2665	0.90
	t cell k_3	27424	0.09
			mod_1 =0.06
$k = 2$	k_1 k_2 response	40143	0.06
	k_1 cell k_3	120056	0.02
	t k_2 k_3	34925	0.07
			mod_2 =0.00008

Table 3: $P\text{-}Mod$ and k-modifiabilities for $k = 1$ and $k = 2$ for the trigram non-term *"t cell response"*

Now, for a particular k ($1 \leq k \leq n$; n = length of n-gram), the frequency of each possible selection, sel, is determined. The paradigmatic modifiability for a particular selection sel is then defined by the n-gram's frequency scaled against the frequency of sel. As can be seen in Tables 2 and 3, a *lower* frequency induces a *more limited* paradigmatic modifiability for a particular sel (which is, of course, expressed as a higher probability value; see the column labeled mod_{sel} in both tables). Thus, with s being the number of distinct possible selections for a particular k, the *k-modifiability*, mod_k, of an n-gram can be defined as follows (f stands for frequency):

$$mod_k(n\text{-}gram) := \prod_{i=1}^{s} \frac{f(n\text{-}gram)}{f(sel_i, n\text{-}gram)} \quad (2)$$

The *paradigmatic modifiability*, $P\text{-}Mod$, of an n-gram is the product of all its k-modifiabilities:[8]

$$P\text{-}Mod(n\text{-}gram) := \prod_{k=1}^{n} mod_k(n\text{-}gram) \quad (3)$$

Comparing the trigram $P\text{-}Mod$ values for $k = 1, 2$ in Tables 2 and 3, it can be seen that the term "*long terminal repeat*" gets a much higher weight than the non-term "*t cell response*", although their mere frequency values suggest the opposite. This is also reflected in the respective list rank (see Subsection 4.1 for details) assigned to both trigrams by the t-test and by our $P\text{-}Mod$ measure. While "*t cell response*" has rank 24 on the t-test output list (which directly reflects its high frequency), $P\text{-}Mod$ assigns rank 1249 to it. Conversely, "*long terminal repeat*" is ranked on position 242 by the t-test, whereas it occupies rank 24 for $P\text{-}Mod$. In fact, even lower-frequency multi-word units gain a prominent ranking, if they exhibit limited paradigmatic modifiability. For example, the trigram term "*porphyria cutanea tarda*" is ranked on position 28 by $P\text{-}Mod$, although its frequency is only 48 (which results in rank 3291 on the t-test output list). Despite its lower frequency, this term is judged as being relevant for the molecular biology domain.[9] It should be noted that the termhood values (and the corresponding list ranks) computed by $P\text{-}Mod$ also include $k = 3$ and, hence, take into account a reasonable amount of frequency load. As can be seen from the previous ranking examples, still this factor does not override the paradigmatic modifiability factors of the lower ks.

On the other hand, $P\text{-}Mod$ will also demote true terms in their ranking, if their paradigmatic modifiability is less limited. This is particularly the case if one or more of the tokens of a particular term often occur in the same slot of other equal-length n-grams. For example, the trigram term *"bone marrow cell"* occurs 1757 times in our corpus and is thus ranked quite high (position 31) by the t-test. $P\text{-}Mod$, however, ranks this term on position 550 because the to-

[8]Setting the upper limit of k to n (e.g., $n = 3$ for trigrams) actually has the pleasant side effect of including frequency in our modifiability measure. In this case, the only possible selection $k_1k_2k_3$ as the denominator of Formula (2) is equivalent to summing up the frequencies of all trigram term candidates.

[9]It denotes a group of related disorders, all of which arise from a deficient activity of the heme synthetic enzyme uroporphyrinogen decarboxylase (URO-D) in the liver.

ken *"cell"* also occurs in many other trigrams and thus leads to a less limited paradigmatic modifiability. Still, the underlying assumption of our approach is that such a case is more an exception than the rule and that terms are linguistically more 'frozen' than non-terms, which is exactly the intuition behind our measure of limited paradigmatic modifiability.

3.4 Methods of Evaluation

As already described in Subsection 3.2, standard procedures for evaluating the quality of termhood measures usually involve identifying the true positives among a (usually) arbitrarily set number of the m highest ranked candidates returned by a particular measure, a procedure usually carried out by a domain expert. Because this is labor-intensive (besides being unreliable), m is usually small, ranging from 50 to several hundreds.[10] By contrast, we choose a large and already consensual terminology to identify the true terms in our candidate sets. Thus, we are able to dynamically examine various m-highest ranked samples, which, in turn, allows for the plotting of standard precision and recall graphs for the entire candidate set. We thus provide a more reliable evaluation setting for ATR measures than what is common practice in the literature.

We compare our $P\text{-}Mod$ algorithm against the t-test measure,[11] which, of all standard measures, yields the best results in general-language collocation extraction studies (Evert and Krenn, 2001), and also against the widely used C-value, which aims at enhancing the common frequency of occurrence measure by making it sensitive to nested terms (Frantzi et al., 2000). Our baseline is defined by the proportion of true positives (i.e., the proportion of terms) in our bi-, tri- and quadgram candidate sets. This is equivalent to the likelihood of finding a true positive by blindly picking from one of the different sets (see Subsection 3.2).

[10]Studies on collocation extraction (e.g., by Evert and Krenn (2001)) also point out the inadequacy of such evaluation methods. In essence, they usually lead to very superficial judgments about the measures under scrutiny.

[11]Manning and Schütze (1999) describe how this measure can be used for the extraction of multi-word expressions.

4 Results and Discussion

4.1 Precision/Recall for Terminology Extraction

For each of the different candidate sets, we incrementally examined portions of the ranked output lists returned by each of the three measures we considered. The precision values for the various portions were computed such that for each percent point of the list, the number of true positives found (i.e., the number of terms) was scaled against the overall number of candidate items returned. This yields the (descending) precision curves in Figures 1, 2 and 3 and some associated values in Table 4.

	Portion of ranked list considered	**Precision** scores of measures		
		$P\text{-}Mod$	t-test	C-value
Bigrams	1%	0.82	0.62	0.62
	10%	0.53	0.42	0.41
	20%	0.42	0.35	0.34
	30%	0.37	0.32	0.31
	baseline	0.22	0.22	0.22
Trigrams	1%	0.62	0.55	0.54
	10%	0.37	0.29	0.28
	20%	0.29	0.23	0.23
	30%	0.24	0.20	0.19
	baseline	0.12	0.12	0.12
Quadgrams	1%	0.43	0.50	0.50
	10%	0.26	0.24	0.23
	20%	0.20	0.16	0.16
	30%	0.18	0.14	0.14
	baseline	0.08	0.08	0.08

Table 4: Precision scores for biomedical term extraction at selected portions of the ranked list

First, we observe that, for the various n-gram candidate sets examined, all measures outperform the baselines by far, and, thus, all are potentially useful measures for grading termhood. Still, the $P\text{-}Mod$ criterion substantially outperforms all other measures at almost all points for all n-grams examined. Considering 1% of the bigram list (i.e., the first 673 candidates) precision for $P\text{-}Mod$ is 20 points higher than for the t-test and the C-value. At 1% of the trigram list (i.e., the first 310 candidates), $P\text{-}Mod$'s lead is 7 points. Considering 1% of the quadgrams (i.e., the first 108 candidates), the t-test actually leads by 7 points. At 10% of the quadgram list, however, the $P\text{-}Mod$ precision score has overtaken the other ones. With increasing portions of all ranked lists considered, the precision curves start to converge toward the baseline, but $P\text{-}Mod$ maintains a steady advantage.

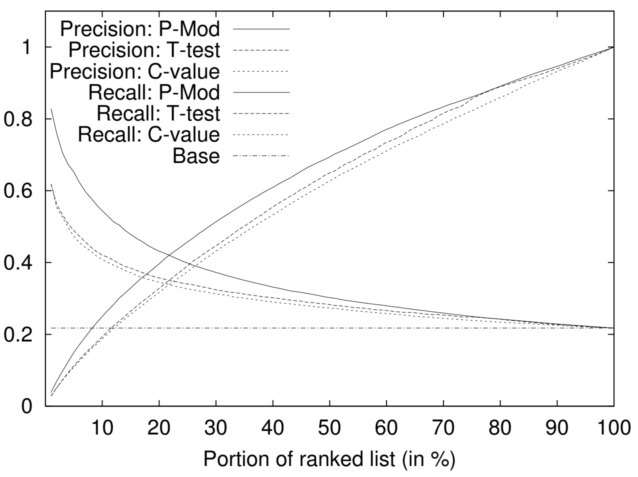


Figure 1: Precision/Recall for bigram biomedical term extraction

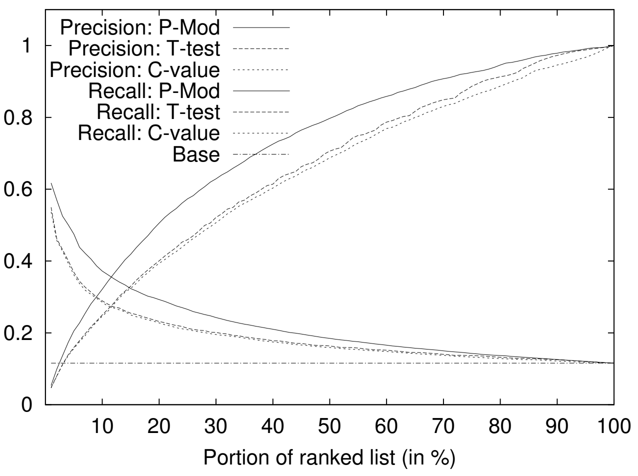


Figure 2: Precision/Recall for trigram biomedical term extraction

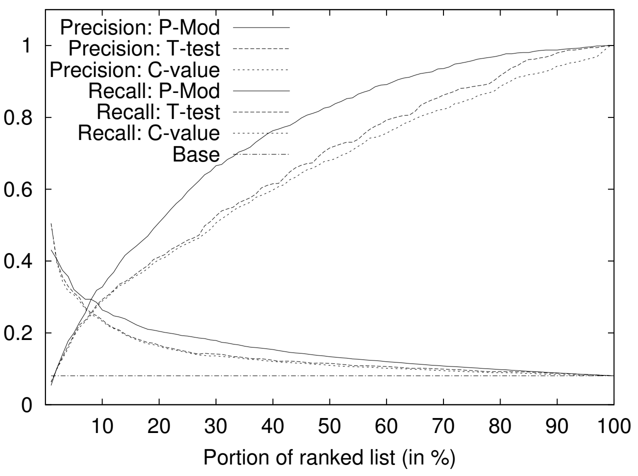


Figure 3: Precision/Recall for quadgram biomedical term extraction

The (ascending) recall curves in Figures 1, 2 and 3 and their corresponding values in Table 5 indicate which *proportion of all true positives* (i.e., the proportion of all terms in a candidate set) is identified by a particular measure at a certain point of the ranked list. For term extraction, recall is an even better indicator of a particular measure's performance because finding a bigger proportion of the true terms at an early stage is simply more economical.

	Recall scores of measures	Portion of Ranked List		
		$P\text{-}Mod$	t-test	C-value
Bigrams	0.5	29%	35%	37%
	0.6	39%	45%	47%
	0.7	51%	56%	59%
	0.8	65%	69%	72%
	0.9	82%	83%	85%
Trigrams	0.5	19%	28%	30%
	0.6	27%	38%	40%
	0.7	36%	50%	53%
	0.8	50%	63%	66%
	0.9	68%	77%	84%
Quadgrams	0.5	20%	28%	30%
	0.6	26%	38%	40%
	0.7	34%	49%	53%
	0.8	45%	62%	65%
	0.9	61%	79%	82%

Table 5: Portions of the ranked list to consider for selected recall scores for biomedical term extraction

Again, our linguistically motivated terminology extraction algorithm outperforms its competitors, and with respect to tri- and quadgrams, its gain is even more pronounced than for precision. In order to get a 0.5 recall for bigram terms, $P\text{-}Mod$ only needs to winnow 29% of the ranked list, whereas the t-test and C-value need to winnow 35% and 37%, respectively. For trigrams and quadgrams, $P\text{-}Mod$ only needs to examine 19% and 20% of the list, whereas the other two measures have to scan almost 10 additional percentage points. In order to obtain a 0.6, 0.7, 0.8 and 0.9 recall, the differences between the measures narrow for bigram terms, but they widen substantially for tri- and quadgram terms. To obtain a 0.6 recall for trigram terms, $P\text{-}Mod$ only needs to winnow 27% of its output list while the t-test and C-value must consider 38% and 40%, respectively. For a level of 0.7 recall, $P\text{-}Mod$ only needs to analyze 36%, while the t-test already searches 50% of the ranked list. For 0.8 recall, this relation is 50%

(*P-Mod*) to 63% (t-test), and at recall point 0.9, 68% (*P-Mod*) to 77% (t-test). For quadgram term identification, the results for *P-Mod* are equally superior to those for the other measures, and at recall points 0.8 and 0.9 even more pronounced than for trigram terms.

We also tested the significance of differences for these results, both comparing *P-Mod vs.* t-test and *P-Mod vs.* C-value. Because in all cases the ranked lists were taken from the same set of candidates (*viz.* the set of bigram, trigram, and quadgram candidate types), and hence constitute dependent samples, we applied the McNemar test (Sachs, 1984) for statistical testing. We selected 100 measure points in the ranked lists, one after each increment of one percent, and then used the two-tailed test for a confidence interval of 95%. Table 6 lists the number of significant differences for these measure points at intervals of 10 for the bi-, tri-, and quadgram results. For the bigram differences between *P-Mod* and C-value, all of them are significant, and between *P-Mod* and t-test, all are significantly different up to measure point 70.[12] Looking at the tri- and quadgrams, although the number of significant differences is less than for bigrams, the vast majority of measure points is still significantly different and thus underlines the superior performance of the *P-Mod* measure.

# of measure points	# of significant differences comparing *P-Mod* with					
	t-test	C-val	t-test	C-val	t-test	C-val
10	10	10	9	9	3	3
20	20	20	19	19	13	13
30	30	30	29	29	24	24
40	40	40	39	39	33	33
50	50	50	49	49	43	43
60	60	60	59	59	53	53
70	70	70	69	69	63	63
80	75	80	79	79	73	73
90	84	90	89	89	82	83
100	93	100	90	98	82	91
	bigrams		trigrams		quadgrams	

Table 6: Significance testing of differences for bi-, tri- and quadgrams using the two-tailed McNemar test at 95% confidence interval

[12]As can be seen in Figures 1, 2 and 3, the curves start to merge at the higher measure points and, thus, the number of significant differences decreases.

4.2 Domain Independence and Corpus Size

One might suspect that the results reported above could be attributed to the corpus size. Indeed, the text collection we employed in this study is rather large (104 million words). Other text genres and domains (e.g., clinical narratives, various engineering domains) or even more specialized biological subdomains (e.g., plant biology) do not offer such a plethora of free-text material as the molecular biology domain. To test the effect a drastically shrunken corpus size might have, we assessed the terminology extraction methods for trigrams on a much smaller-sized subset of our original corpus, *viz.* on 10 million words. These results are depicted in Figure 4.

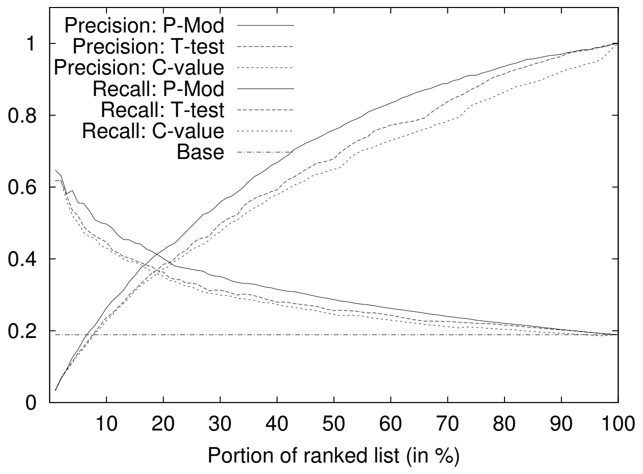


Figure 4: Precision/Recall for trigram biomedical term extraction on the 10-million-word corpus (cutoff $c \geq 4$, with 6,760 term candidate types)

The *P-Mod* extraction criterion still clearly outperforms the other ones on that 10-million-word corpus, both in terms of precision and recall. We also examined whether the differences were statistically significant and applied the two-tailed McNemar test on 100 selected measure points. Comparing *P-Mod* with t-test, most significant differences could be observed between measure points 20 and 80, with almost 80% to 90% of the points being significantly different. These significant differences were even more pronounced when comparing the results between *P-Mod* and C-value.

5 Conclusions

We here proposed a new terminology extraction method and showed that it significantly outperforms two of the standard approaches in distinguishing terms from non-terms in the biomedical literature. While mining scientific literature for new terminological units and assembling those in controlled vocabularies is a task involving several components, one essential building block is to measure the *degree of termhood* of a candidate. In this respect, our study has shown that a criterion which incorporates a vital linguistic property of terms, *viz.* their *limited paradigmatic modifiability*, is much more powerful than linguistically more uninformed measures. This is in line with our previous work on general-language collocation extraction (Wermter and Hahn, 2004), in which we showed that a linguistically motivated criterion based on the limited *syntagmatic* modifiability of collocations outperforms alternative standard association measures as well.

We also collected evidence that the superiority of the $P\text{-}Mod$ method relative to other term extraction approaches holds independent of the underlying corpus size (given a reasonable offset). This is a crucial finding because other domains might lack large volumes of free-text material but still provide sufficient corpus sizes for valid term extraction. Finally, since we only require shallow syntactic analysis (in terms of NP chunking), our approach might be well suited to be easily portable to other domains. Hence, we may conclude that, although our methodology has been tested on the biomedical domain only, there are essentially no inherent domain-specific restrictions.

Acknowledgements. This work was partly supported by the European Network of Excellence "Semantic Mining in Biomedicine" (NoE 507505).

References

Olivier Bodenreider, Thomas C. Rindflesch, and Anita Burgun. 2002. Unsupervised, corpus-based method for extending a biomedical terminology. In Stephen Johnson, editor, *Proceedings of the ACL/NAACL 2002 Workshop on 'Natural Language Processing in the Biomedical Domain'*, pages 53–60. Philadelphia, PA, USA, July 11, 2002.

Nigel Collier, Chikashi Nobata, and Jun'ichi Tsujii. 2002. Automatic acquisition and classification of terminology using a tagged corpus in the molecular biology domain. *Terminology*, 7(2):239–257.

Béatrice Daille. 1996. Study and implementation of combined techniques for automatic extraction of terminology. In Judith L. Klavans and Philip Resnik, editors, *The Balancing Act: Combining Statistical and Symbolic Approaches to Language*, pages 49–66. Cambridge, MA: MIT Press.

Fred J. Damerau. 1993. Generating and evaluating domain-oriented multi-word terms from text. *Information Processing & Management*, 29(4):433–447.

Stefan Evert and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In *ACL'01/EACL'01 – Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and the 10th Conference of the European Chapter of the ACL*, pages 188–195. Toulouse, France, July 9-11, 2001.

Katerina T. Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms: The C-value/NC-value method. *International Journal on Digital Libraries*, 3(2):115–130.

Christian Jacquemin. 1999. Syntagmatic and paradigmatic representations of term variation. In *Proceedings of the 37rd Annual Meeting of the Association for Computational Linguistics*, pages 341–348. College Park, MD, USA, 20-26 June 1999.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL'01, Language Technologies 2001 – Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 192–199. Pittsburgh, PA, USA, June 2-7, 2001.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA; London, U.K.: Bradford Book & MIT Press.

Hiroshi Nakagawa and Tatsunori Mori. 2002. A simple but powerful automatic term extraction method. In COMPUTERM *2002 – Proceedings of the 2nd International Workshop on Computational Terminology*, pages 29–35. Taipei, Taiwan, August 31, 2002.

Goran Nenadić, Irena Spasić, and Sophia Ananiadou. 2003. Terminology-driven mining of biomedical literature. *Bioinformatics*, 19(8):938–943.

Goran Nenadić, Sophia Ananiadou, and John McNaught. 2004. Enhancing automatic term recognition through recognition of variation. In *COLING 2004 – Proceedings of the 20th International Conference on Computational Linguistics*, pages 604–610. Geneva, Switzerland, August 23-27, 2004.

Thomas C. Rindflesch, Lawrence Hunter, and Alan R. Aronson. 1999. Mining molecular binding terminology from biomedical text. In *AMIA'99 – Proceedings of the Annual Symposium of the American Medical Informatics Association*, pages 127–131. Washington, D.C., November 6-10, 1999.

Lothar Sachs. 1984. *Applied Statistics: A Handbook of Techniques*. New York: Springer, 2nd edition.

MESH. 2004. *Medical Subject Headings*. Bethesda, MD: National Library of Medicine.

UMLS. 2004. *Unified Medical Language System*. Bethesda, MD: National Library of Medicine.

Joachim Wermter and Udo Hahn. 2004. Collocation extraction based on modifiability statistics. In *COLING 2004 – Proceedings of the 20th International Conference on Computational Linguistics*, pages 980–986. Geneva, Switzerland, August 23-27, 2004.

A Backoff Model for Bootstrapping Resources for Non-English Languages*

Chenhai Xi and Rebecca Hwa
Department of Computer Science
University of Pittsburgh
{chenhai,hwa}@cs.pitt.edu

Abstract

The lack of annotated data is an obstacle to the development of many natural language processing applications; the problem is especially severe when the data is non-English. Previous studies suggested the possibility of acquiring resources for non-English languages by bootstrapping from high quality English NLP tools and parallel corpora; however, the success of these approaches seems limited for dissimilar language pairs. In this paper, we propose a novel approach of combining a bootstrapped resource with a small amount of manually annotated data. We compare the proposed approach with other bootstrapping methods in the context of training a Chinese Part-of-Speech tagger. Experimental results show that our proposed approach achieves a significant improvement over EM and self-training and systems that are only trained on manual annotations.

1 Introduction

Natural language applications that use supervised learning methods require annotated training data, but annotated data is scarce for many non-English languages. It has been suggested that annotated data for these languages might be automatically created by leveraging parallel corpora and high-accuracy English systems (Yarowsky and Ngai, 2001; Diab and Resnik, 2002). The studies are centered around the assumption that linguistic analyses for English (e.g., Part-of-Speech tags, Word sense disambiguation, grammatical dependency relationships) are also valid analyses in the translation of the English. For example, in the English noun phrase *the red apples*, *red* modifies *apples*; the same modifier relationship also exists in its French translations *les pommes rouges*, even though the word orders differ. To the extent that the assumption is true, annotated data in the non-English language can be created by projecting English analyses across a word aligned parallel corpus. The resulting projected data can then serve as (albeit noisy) training examples to develop applications in the non-English language.

The projection approach faces both a theoretical and a practical challenge. Theoretically, it is well-known that two languages often do not express the same meaning in the same way (Dorr, 1994). Practically, the projection framework is sensitive to component errors. In particular, poor word alignments significantly degrade the accuracy of the projected annotations. Previous research on resource projection attempts to address these problems by redistributing the parameter values (Yarowsky and Ngai, 2001) or by applying transformation rules (Hwa et al.,

*We thank Stephen Clark, Roger Levy, Carol Nichols, and the three anonymous reviewers for their helpful comments.

2002). Their experimental results suggest that while these techniques can overcome some errors, they are not sufficient for projected data that are very noisy.

In this work, we tackle the same problems by relaxing the *zero manual annotation* constraint. The main question we address is: how can we make the most out of a small set of manually labeled data (on the non-English side). Following the work of Yarowsky and Ngai (2001) we focus on the task of training a Part-of-Speech (POS) tagger, but we conduct our experiments with the more dissimilar language pair of English-Chinese instead of English-French. Through empirical studies, we show that when the word alignment quality is sufficiently poor, the error correction techniques proposed by Yarowsky and Ngai are unable to remove enough mistakes in the projected data. We propose an alternative approach that is inspired by backoff language modeling techniques in which the parameters of two tagging models (one trained on manually labeled data; the other trained on projected data) are combined to achieve a more accurate final model.

2 Background

The idea of trying to squeeze more out of annotated training examples has been explored in a number of ways in the past. Most popular is the family of bootstrapping algorithms, in which a model is seeded with a small amount of labeled data and iteratively improved as more unlabeled data are folded into the training set, typically, through unsupervised learning. Another approach is *active learning* (Cohn et al., 1996), in which the model is also iteratively improved but the training examples are chosen by the learning model, and the learning process is supervised. Finally, the work that is the closest to ours in spirit is the idea of joint estimation (Smith and Smith, 2004).

Of the bootstrapping methods, perhaps the most well-known is the Expectation Maximization (EM) algorithm. This approach has been explored in the context of many NLP applications; one example is text classification (Nigam et al., 1999). Another bootstrapping approach reminiscent of EM is *self-training.* Yarowsky (1995) used this method for word sense disambiguation. In self-training, annotated examples are used as seeds to train an initial classifier with any supervised learning method. This initial classifier is then used to automatically annotate data from a large pool of unlabeled examples. Of these newly labeled data, the ones labeled with the highest confidence are used as examples to train a new classifier. Yarowsky showed that repeated application of this process resulted in a series of word sense classifiers with improved accuracy and coverage. Also related is the co-training algorithm (Blum and Mitchell, 1998) in which the bootstrapping process requires multiple learners that have different *views* of the problem. The key to co-training is that the views should be *conditionally independent* given the label. The strong independence requirement on the views is difficult to satisfy. For practical applications, different features sets or models (that are not conditionally independent) have been used as an approximation for different views. Co-training has been applied to a number of NLP applications, including POS-tagging (Clark et al., 2003), parsing (Sarkar, 2001), word sense disambiguation (Mihalcea, 2004), and base noun phrase detection (Pierce and Cardie, 2001). Due to the relaxation of the view independence assumption, most empirical studies suggest a marginal improvement. The common thread between EM, self-training, and co-training is that they all bootstrap off of unannotated data. In this work, we explore an alternative to "pure" unannotated data; our data have been automatically annotated with projected labels from English. Although the projected labels are error-prone, they provide us with more information than automatically predicted labels used in bootstrapping methods.

With a somewhat different goal in mind, active learning addresses the problem of choosing the most informative data for annotators to label so that the model would achieve the greatest improvement. Active learning also has been applied to many NLP applications, including POS tagging (Engelson and Dagan, 1996) and parsing (Baldridge and Osborne, 2003). The drawback of an active learning approach is that it assumes that a staff of annotators is waiting on call, ready to label the examples chosen by the system at every iteration. In practice, it is more likely that one could only afford to staff annotators for a limited period of time. Although active learning is not a focus in this paper, we owe some ideas to active learning in choosing a small initial set of training examples; we discuss these ideas in section 3.2.

More recently, Smith and Smith (2004) proposed to merge an English parser, a word alignment model, and a Korean PCFG parser trained from a small number of Korean parse trees under a unified log linear model. Their results suggest that a joint model produces somewhat more accurate Korean parses than a PCFG Korean parser trained on a small amount of annotated Korean parse trees alone. Their motivation is similar to the starting point of our work: that a word aligned parallel corpus and a small amount of annotated data in the foreign language side offer information that might be exploited. Our approach differs from theirs in that we do not optimize the three models jointly. One concern is that joint optimization might not result in optimal parameter settings for the individual components. Because our focus is primarily on acquiring non-English language resources, we only use the parallel corpus as a means of projecting resources from English.

3 Our Approach

This work explores developing a Chinese POS tagger without a large manually annotated corpus. Our approach is to train two separate models from two different data sources: a large corpus of automatically tagged data (projected from English) and a small corpus of manually tagged data; the two models are then combined into one via the Whitten-Bell backoff language model.

3.1 Projected Data

One method of acquiring a large corpus of automatically POS tagged Chinese data is by *projection* (Yarowsky and Ngai, 2001). This approach requires a sentence-aligned English-Chinese corpus, a high-quality English tagger, and a method of aligning English and Chinese words that share the same meaning. Given the parallel corpus, we tagged the English words with a publicly available maximum entropy tagger (Ratnaparkhi, 1996), and we used an implementation of the IBM translation model (Al-Onaizan et al., 1999) to align the words. The Chinese words in the parallel corpus would then receive the same POS tags as the English words to which they are aligned. Next, the basic projection algorithm is modified to accommodate two complicating factors. First, word alignments are not always one-to-one. To compensate, we assign a default tag to unaligned Chinese words; in the case of one-Chinese-to-many-English, the Chinese word would receive the tag of the final English word. Second, English and Chinese do not share the same tag set. Following Yarowsky and Ngai (2001), we define 12 equivalence classes over the 47 Penn-English-Treebank POS tags. We refer to them as *Core Tags*. With the help of 15 hand-coded rules and a Naive Bayes model trained on a small amount of manually annotated data, the Core Tags can be expanded to the granularity of the 33 Penn-Chinese-Treebank POS tags (which we refer to as *Full Tags*).

3.2 Manually Annotated Data

Since the amount of manual annotation is limited, we must decide what type of data to annotate. In the spirit of active learning, we aim to select sentences that may bring about the greatest improvements in the accuracy of our model. Because it is well known that handling unknown words is a serious problem for POS taggers, our strategy for selecting sentences for manual annotation is to maximize the word coverage of the in ital model. That is, we wish to find a small set of sentences that would lead to the greatest reduction of currently unknown words Finding these sentences is a NP-hard problem because the 0/1 knapsack problem could be reduced to this problem in polynomial-time (Gurari, 1989). Thus, we developed an approximation algorithm for finding sentences with the maximum word

```
$M$ : number of tokens will be annotated.
$S$={$s_1, s_2, \ldots, s_n$}: the unannotated corpus.
$S_{sel}$ : set of selected sentences in $S$.
$S_{unsel}$ : set of unselected sentences in $S$.
$|S_{sel}|$ : number of tokens in $S_{sel}$.
$TYPE(S_{sel})$ : number of types in $S_{sel}$.
**MWC:**
    randomly choose $S_{sel} \subset S$
     such that $|S_{sel}| \leq M$.
 **For** each sentence $s_i$ in $S_{sel}$
    find a sentence $r_j$ in $S_{unsel}$
      which maximizes **swap_score**($s_i, r_j$).
    if **swap_score**($s_i, r_j$) $> 0$
    {
        $S_{sel} = (S_{sel} - s_i) \cup r_j$;
        $S_{unsel} = (S_{unsel} - r_j) \cup s_i$;
    }

**swap_score**($s_i, r_j$)
{
   $S_{sel\_new} = (S_{sel} - s_i) \cup r_j$;
   if ( $|S_{sel\_new}| > M$ ) return -1;
   else return $TYPE(S_{sel\_new}) - TYPE(S_{sel})$;
}
```

Figure 1: The pseudo-code for MWC algorithm. The input is M and S and the output is S_{sel}

coverage of unknown words (MWC). This algorithm is described in Figure 1,

3.3 Basic POS Tagging Model

It is well known that a POS tagger can be trained with an HMM (Weischedel et al., 1993). Given a trained model, the most likely tag sequence $\hat{T} = \{t_1, t_2, \ldots t_n\}$ is computed for the input word sentence: $\hat{W} = \{w_1, w_2, \ldots w_n\}$:

$$\hat{T} = \arg\max_T P(T|W) = \arg\max_T P(T|W)P(T)$$

The transition probability $P(T)$ is approximated by a trigram model:

$$P(T) \approx p(t_1)p(t_2|t_1)\prod_{i=3}^{n} p(t_i|t_{i-1}, t_{i-2}),$$

and the observation probability $P(W|T)$ is computed by

$$P(W|T) \approx \prod_{i=1}^{n} p(w_i|t_i).$$

3.4 Combined Models

From the two data sources, two separate trigram taggers have been trained (T_{anno} from manually annotated data and T_{proj} from projected data). This section considers ways of combining them into a single tagger. The key insight that drives our approach is based on reducing the effect of unknown words. We see the two data sources as complementary in that the large projected data source has better word coverage while the manually labeled one is good at providing tag-to-tag transitions. Based on this principle, one way of merging these two taggers into a single HMM (denoted as T_{interp}) is to use interpolation:

$$\begin{aligned} p_{interp}(w|t) &= \lambda \times p_{anno}(w|t) \\ &\quad +(1-\lambda) \times p_{proj}(w|t) \\ p_{interp}(t_i|t_{i-1}, t_{i-2}) &= p_{anno}(t_i|t_{i-1}, t_{i-2}) \end{aligned}$$

where λ is a tunable weighting parameter[1] of the merged tagger. This approach may be problematic because it forces the model to always include some fraction of poor parameter values. Therefore, we propose to estimate the observation probabilities using backoff. The parameters of T_{back} are estimated as follows:

$$p_{back}(w|t) = \begin{cases} \alpha(t) \times p_{anno}(w|t) \text{ if } p_{anno}(w|t) > 0 \\ \beta(t) \times p_{proj}(w|t) \text{ if } p_{anno}(w|t) = 0 \end{cases}$$

$$p_{back}(t_i|t_{i-1}, t_{i-2}) = p_{anno}(t_i|t_{i-1}, t_{i-2})$$

where $\alpha(t)$ is a discounting coefficient and β is set to satisfy that $\sum_{\text{all words}} P(w|t) = 1$. The discounting coefficient is computed using the Witten-Bell discounting method:

$$\alpha(t) = \frac{C_{anno}(t)}{C_{anno}(t) + S_{anno}(t)},$$

where $C_{anno}(t)$ is the count of tokens whose tag is t in the manually annotated corpus and

[1]In our experiments, the value of λ is set to 0.8 based on held-out data.

$S_{anno}(t)$ is the seen types of words with tag t. In other words, we trust the parameter estimates from the model trained on manual annotation by default unless it is based on unreliable statistics.

4 Experiments

We conducted a suite of experiments to investigate the effect of allowing a small amount of manually annotated data in conjunction with using annotations projected from English. We first establish a baseline of training on projected data alone in Section 4.1. It is an adaptation of the approach described by Yarowsky and Ngai (2001). Next, we consider the case of using manually annotated data alone in Section 4.2. We show that there is an increase in accuracy when the MWC active learning strategy is used. In Section 4.3, we show that with an appropriate merging strategy, a tagger trained from both data sources achieves higher accuracy. Finally, in Section 4.4, we evaluate our approach against other semi-supervised methods to verify that the projected annotations, though noisy, contain useful information.

We use an English-Chinese Federal Broadcast Information Service (FBIS) corpus as the data source for the projected annotation. We simulated the manual annotation process by using the POS tags provided by the Chinese Treebank version 4 (CHTB). We used about a thousand sentences from CHTB as held-out data. The remaining sentences are split into ten-fold cross validation sets. Each test set contains 1400 sentences. Training data are selected (using MWC) from the remaining 12600 sentences. The reported results are the average of the ten trials. One tagger is considered to be better than another if, according to the paired t-test, we are at least 95% confident that their difference in accuracy is non-zero. Performance is measured in terms of the percentage of correctly tagged tokens in the test data. For comparability with T_{proj} (which assumes no availability of manually annotated data), most experimental results are reported with respect to the reduced Core Tag gold standard; evaluation against the full 33 CHTB tag gold standard is reported in Section 4.4.

4.1 Tagger Trained from Projected Data

To determine the quality of T_{proj} for Chinese, we replicate the POS-tagging experiment in Yarowsky and Ngai (2001). Trained on all projected data, the tagger has an accuracy of 58.2% on test sentences. The low accuracy rate suggests that the projected data is indeed very noisy. To reduce the noise in the projected data, Yarowsky and Ngai developed a re-estimation technique based on the observation that words in French, English and Czech have a strong tendency to exhibit only a single core POS tag and very rarely have more than two. Applying the same re-estimation technique that favors this bias to the projected Chinese data raises the final tagger accuracy to 59.1%. That re-estimation did not help English-Chinese projection suggests that the dissimilarity between the two languages is an important factor. A related reason for the lower accuracy rate is due to poor word alignments in the English-Chinese corpus. As a further noise reduction step, we automatically filter out sentence pairs that were poorly aligned (i.e., the sentence pairs had too many unaligned words or too many one-to-many alignments). This results in a corpus of about 9000 FBIS sentences. A tagger trained on the filtered data has an improved accuracy of 64.5%. We take this to be T_{proj} used in later experiments.

4.2 Taggers Trained from Manually Labeled Data

This experiment verifies that the Maximum Word Coverage (MWC) selection scheme presented in Section 3.2 is helpful in selecting data for training T_{anno}. We compare it against random selection. Figure 2 plots the taggers' performances on test sentences as the number of manually annotated tokens increase from 100 to 30,000. We see that the taggers trained on data selected by MWC outperform those trained on randomly selected data. Thus, in the main experiments, we always use MWC to select the set of manually tagged data for training T_{anno}.

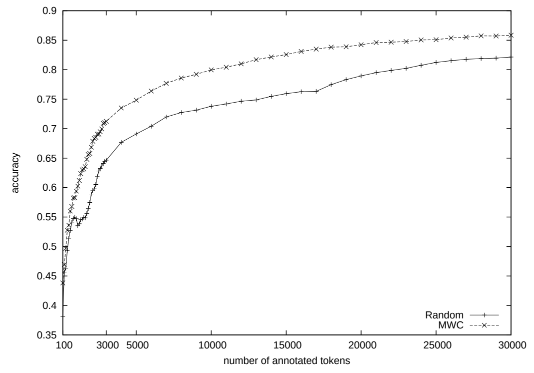


Figure 2: A comparison between MWC and random selection.

4.3 Evaluation of the Combined Taggers

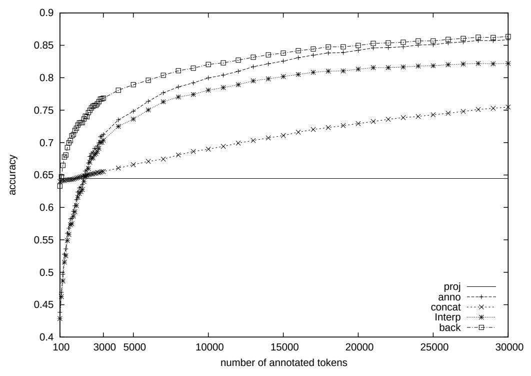


Figure 3: A comparison of the proposed backoff approach against alternative methods of combining T_{proj} and T_{anno}

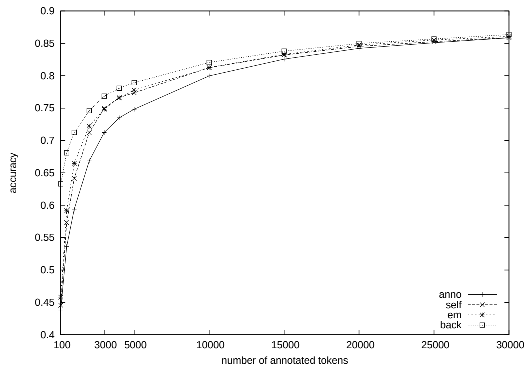
To investigate how T_{anno} and T_{proj} might be merged to form a higher quality tagger, we conduct an experiment to evaluate the different alternatives described in section 3.4: T_{interp}, and T_{back}. They are compared against three baselines: T_{anno}, T_{proj}, and T_{concat}, a tagger trained from the concatenation of the two data sources. To determine the effect of manual annotation, we vary the size of the training set for T_{anno} from 100 tokens (fewer than 10 sentences) to 30,000 tokens (about 1000 sentences). The learning curves are plotted in Figure 3. The result suggests that T_{back} successfully incorporates information from both the manually annotated data and the projected data. The improvement over training on manually annotated data alone (T_{anno}) is especially high when fewer than 10,000 manually annotated tokens are available. As expected, T_{interp}, and T_{concat} perform worse than T_{anno} because they are not as effective at discounting the erroneous projected annotations.

4.4 Comparisons with Other Semi-Supervised Approaches

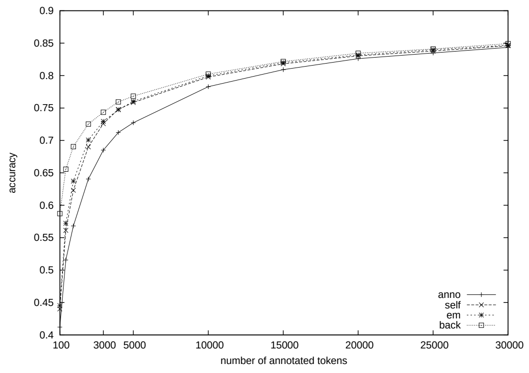
This experiment evaluates the proposed backoff approach against two other semi-supervised approaches: self-training (denoted as T_{self}) and EM (denoted as T_{em}). Both start with a fully supervised model (T_{anno}) and iteratively improve it by seeing more unannotated data.[2] As discussed earlier, a major difference between our proposed approach and the bootstrapping methods is that our approach makes use of annotations projected from English while the bootstrapping methods rely on unannotated data alone. To investigate the effect of leveraging from English resources, we use the Chinese portion of the FBIS parallel corpus (the same 9000 sentences as the training corpus of T_{proj} but without the projected tags) as the unannotated data source for the bootstrapping methods.

Figure 4 compares the four learning curves. We have evaluated them both in terms of the Core Tag gold standard and in terms of Full Tag gold standard. Although all three approaches produce taggers with higher accuracies than that of T_{anno}, our backoff approach outperforms both self-training and EM. The difference is especially prominent when manual annotation is severely limited. When more manual annotations are made available, the gap narrows; however, the differences are still statistically significant at 30,000 manually annotated tokens. These results suggest that projected data have more useful information than unannotated data.

[2]In our implementation of self-training, the top 10% of the unannoated sentences with the highest confidence scores is selected. The confidence score is computed as: $\frac{\log P(T|W)}{\text{length of the sentence}}$.



(a)



(b)

Figure 4: A comparison of Backoff against self-training and EM. (a) Evaluation against the Core Tag gold standard. (b) Evaluation against the Full Tag gold standard.

5 Discussion

While the experimental results support our intuition that T_{back} is effective in making use of both data sources, there are still two questions worth addressing. First, there may be other ways of estimating the parameters of a merged HMM from the parameters of T_{anno} and T_{proj}. For example, a natural way of merging the two taggers into a single HMM (denoted as T_{merge}) is to use the values of the observation probabilities from T_{proj} and the values of the transition probabilities from T_{anno}:

$$\begin{aligned} p_{merge}(w|t) &= p_{proj}(w|t), \\ p_{merge}(t_i|t_{i-1}, t_{i-2}) &= p_{anno}(t_i|t_{i-1}, t_{i-2}). \end{aligned}$$

Another is the reverse of T_{merge}:

$$\begin{aligned} p_{rev\_merge}(w|t) &= p_{anno}(w|t) \\ p_{rev\_merge}(t_i|t_{i-1}, t_{i-2}) &= p_{proj}(t_i|t_{i-1}, t_{i-2}) \end{aligned}$$

T_{merge} is problematic because it ignores all manual word-tag annotations; however, T_{rev_merge}'s learning curve is nearly identical to that of T_{anno} (graph not shown). Its models do not take advantage of the broader word coverage of the projected data, so it does not perform as well as T_{back}. T_{rev_merge} outperforms T_{merge} when trained from more than 2000 manually annotated tokens. We make two observations from this finding. One is that the differences between $p_{proj}(t_i|t_{i-1}, t_{i-2})$ and $p_{anno}(t_i|t_{i-1}, t_{i-2})$ are not large. Another is that the success of the merged HMM tagger hinges on the goodness of the observation probabilities, $p(w|t)$. This is in accord with our motivation in improving the reliability of $p(w|t)$ through backoff.

Second, while our experimental results suggest that T_{back} outperforms self-training and EM, these approaches are not incompatible with one another. Because T_{back} is partially estimated from the noisy corpus of projected annotations, it might be further improved by applying a bootstrapping algorithm over the noisy corpus (with the projected tags removed). To test our hypothesis, we initialized the self-training algorithm with a backoff tagger that used 3000 manually annotated tokens. This led to a slight but statistically significant improvement, from 74.3% to 74.9%.

6 Conclusion and Future Work

In summary, we have shown that backoff is an effective technique for combining manually annotated data with a large but noisy set of automatically annotated data (from projection). Our approach is the most useful when a small amount of annotated tokens is available. In our experiments, the best results were achieved when we used 3000 manually annotated tokens (approximately 100 sentences).

The current study points us to several directions for future work. One is to explore ways of applying the proposed approach to other learning models. Another is to compare against other methods of combining evidences from multiple learners. Finally, we will investigate whether the proposed approach can be adapted to more complex tasks in which the output is not a class label but a structure (e.g. parsing).

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, I. Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation. Technical report, JHU. citeseer.nj.nec.com/alonaizan99statistical.html.

Jason Baldridge and Miles Osborne. 2003. Active learning for HPSG parse selection. In *Proceedings of the 7th Conference on Natural Language Learning*, Edmonton, Canada, June.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, pages 92–100, Madison, WI.

Stephen Clark, James Curran, and Miles Osborne. 2003. Bootstrapping pos-taggers using unlabelled data. In *Proc. of the Computational Natural Language Learning Conference*, pages 164–167, Edmonton, Canada, June.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Bonnie J. Dorr. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4):597–635.

Sean P. Engelson and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from copora. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 319–326, Santa Cruz, CA.

Eitan Gurari. 1989. *An Introduction to the Theory of Computation*. Ohio State University Computer Science Press.

Rebecca Hwa, Philip S. Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004)*.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 1999. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 1(34).

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*, pages 1–9, Pittsburgh, PA.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Association for Computational Linguistics*, pages 175–182, Pittsburgh, PA, June.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP-05)*.

Ralph Weischedel, Richard Schwartz, Jeff Palmucci, Marie Meteer, and Lance Ramshaw. 1993. Coping with ambiguity and unknown words through probabilistic models. *Comput. Linguist.*, 19(2):361–382.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Association for Computational Linguistics*, pages 200–207.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.

Cross-linguistic Projection of Role-Semantic Information

Sebastian Padó
Computational Linguistics
Saarland University
Saarbrücken, Germany
pado@coli.uni-sb.de

Mirella Lapata
School of Informatics
University of Edinburgh
Edinburgh, UK
mlap@inf.ed.ac.uk

Abstract

This paper considers the problem of automatically inducing role-semantic annotations in the FrameNet paradigm for new languages. We introduce a general framework for semantic projection which exploits parallel texts, is relatively inexpensive and can potentially reduce the amount of effort involved in creating semantic resources. We propose projection models that exploit lexical and syntactic information. Experimental results on an English-German parallel corpus demonstrate the advantages of this approach.

1 Introduction

Shallow semantic parsing, the task of automatically identifying the semantic roles conveyed by sentential constituents, has recently attracted much attention, partly because of its increasing importance for potential applications. For instance, information extraction (Surdeanu et al., 2003), question answering (Narayanan and Harabagiu, 2004) and machine translation (Boas, 2002) could stand to benefit from broad coverage semantic processing.

The FrameNet project (Fillmore et al., 2003) has played a central role in this endeavour by providing a large lexical resource based on semantic roles. In FrameNet, meaning is represented by frames, schematic representations of situations. Semantic roles are frame-specific, and are called frame elements. The database associates frames with lemmas (verbs, nouns, adjectives) that can evoke them (called frame-evoking elements or FEEs), lists the possible syntactic realisations of their semantic roles, and provides annotated examples from the British National Corpus (Burnard, 1995). The availability of rich annotations for the surface realisation of semantic roles has triggered interest in semantic parsing and enabled the development of data-driven models (e.g., Gildea and Jurafsky, 2002).

Frame: DEPARTING		
Frame Elements	THEME	**The officer** left the house. **The plane** leaves at seven. **His** departure was delayed.
	SOURCE	We departed **from New York**. He retreated **from his opponent**. The woman left the **house**.
FEEs	abandon.v, desert.v, depart.v, departure.n, emerge.v, emigrate.v, emigration.n, escape.v, escape.n, leave.v, quit.v, retreat.v, retreat.n, split.v, withdraw.v, withdrawal.n	

Table 1: Example of FrameNet frame

Table 1 illustrates an example from the FrameNet database, the DEPARTING frame. It has two roles, a THEME which is the moving object and a SOURCE expressing the initial position of the THEME. The frame elements are realised by different syntactic expressions. For instance, the THEME is typically an NP, whereas the SOURCE is often expressed by a prepositional phrase (see the expressions in boldface in Table 1). The DEPARTING frame can be evoked by *abandon*, *desert*, *depart*, and several other verbs as well as nouns (see the list of FEEs in Table 1).

Although recent advances in semantic parsing[1] have greatly benefited from the availability of the English FrameNet, unfortunately such resources are largely absent for other languages. The English FrameNet (Version 1.1) contains 513 frames covering 7,125 lexical items and has been under development for approximately six years. Although FrameNets are currently under construction for German, Spanish, and Japanese, these resources are still in their infancy and of limited value for modelling purposes. Methods for acquiring FrameNets from corpora automatically would greatly reduce the human effort involved and facilitate their development for new languages.

In this paper, we propose a method which employs parallel corpora for acquiring frame elements

[1]Approaches to modelling semantic parsing are too numerous to list; see Carreras and Màrquez (2005) for an overview.

and their syntactic realisations (see the upper half of Table 1) for new languages. Our method leverages the existing English FrameNet to overcome the resource shortage in other languages by exploiting the translational and structural equivalences present in aligned data. The idea underlying our approach can be summarised as follows: (1) given a pair of sentences *E* (English) and *L* (new language) that are translations of each other, annotate *E* with semantic roles; and then (2) *project* these roles onto *L*. In this manner, we induce semantic structure on the *L* side of the parallel text, which can then serve as data for training a statistical semantic parser for *L* that is independent of the parallel corpus.

We first assess if the main assumption of semantic projection is warranted (Section 3), namely whether frames and semantic roles exhibit a high degree of parallelism across languages. Then we propose two broad classes of projection models that utilise lexical and syntactic information (Section 4), and show experimentally that roles can be projected from English onto German with high accuracy (Section 5). We conclude the paper by discussing the implications of our results and future work (Section 6).

2 Related work

A number of recent studies exploit parallel corpora for cross-linguistic knowledge induction. In this paradigm, annotations for resource-rich languages like English are projected onto another language through aligned parallel texts. Yarowsky et al. (2001) propose several projection algorithms for deriving monolingual tools (ranging from part-of-speech taggers, to chunkers and morphological analysers) without additional annotation cost. Hwa et al. (2002) assess the degree of syntactic parallelism in dependency relations between English and Chinese. Their results show that, although assuming direct correspondence is often too restrictive, syntactic projection yields good enough annotations to train a dependency parser. Smith and Smith (2004) explore syntactic projection further by proposing an English-Korean bilingual parser integrated with a word translation model.

Previous work has primarily focused on the projection of morphological and grammatico-syntactic information. Inducing semantic resources from low density languages still poses a significant challenge to data-driven methods. The challenge is recognised by Fung and Chen (2004) who construct a Chinese FrameNet by mapping English FrameNet entries to concepts listed in HowNet[2], an on-line ontology for Chinese, however without exploiting parallel texts.

The present work extends previous approaches on annotation projection by inducing FrameNet semantic roles from parallel corpora. Analogously to Hwa et al. (2002), we investigate whether there are indeed semantic correspondences between two languages, since there is little hope for projecting meaningful annotations in nonparallel semantic structures. Similarly to Fung and Chen (2004) we automatically induce semantic role annotations for a target language. In contrast to them, we resort to parallel corpora as a source of semantic equivalence. Thus, we avoid the need for a target concept dictionary in addition to the English FrameNet. We propose a general framework for semantic projection that can incorporate different knowledge sources. To our knowledge, the framework and its application to semantic role projection are novel.

3 Creation of a Gold Standard Corpus

Sample Selection. To evaluate the output of our projection algorithms, we created a gold standard corpus of English-German sentence pairs with manual FrameNet frame and role annotations. The sentences were sampled from Europarl (Koehn, 2002), a corpus of professionally translated proceedings of the European Parliament. Europarl is available in 11 languages with up to 20 million words per language aligned at the document and sentence level.

Recall that frame projection is only meaningful if the same frame is appropriate for both sentences in a projection pair. This constrains sample selection for two reasons: first, FrameNet is as yet incomplete with respect to its coverage. So, a randomly selected sentence pair may evoke novel frames or novel senses of already existing frames (e.g., the "greeting" sense of *hail* which is currently not listed in FrameNet). Second, due to translational variance, there is no a priori guarantee that words which are mutual translations evoke the same frame. For example, the English verb *finish* is often translated in German by the adverb *abschließend*, which arguably cannot have a role set identical to *finish*. Relying solely on the English FrameNet database for sampling would yield many sentence pairs which are either inappropriate for the present study (because they do not evoke the same frames) or simply problematic for annotation since they are outside the

[2]See `http://www.keenage.com/zhiwang/e_zhiwang.html`.

present coverage of the database.

For the above reasons, our sample selection procedure was informed by two existing resources, the English FrameNet and SALSA, a FrameNet-compatible database for German currently under development (Erk et al., 2003). We first used the publicly available GIZA++ (Och and Ney, 2003) software to induce English-German word alignments. Next, we gathered all German-English sentences in the corpus that had at least one pair of aligned words (w_e, w_g), which were listed in FrameNet and SALSA, respectively, and had at least one frame in common. These sentences exemplify 83 frame types, 696 lemma pairs, and 265 unique English and 178 unique German lemmas. Sentence pairs were grouped into three bands according to their frame frequency (High, Medium, Low). We randomly selected 380 pairs from each band. The total sample consisted of ,140 sentence pairs.

This procedure produces a realistic corpus sample for the role projection task; similar samples can be drawn for new language pairs using either existing bilingual dictionaries (Fung and Chen, 2004) or automatically constructed semantic lexicons (Padó and Lapata, 2005).

Annotation. Two annotators, with native-level proficiency in German and English, manually labelled the parallel corpus with semantic information. Their task was to identify the frame for a given predicate in a sentence, and assign the corresponding roles. They were provided with detailed guidelines that explained the task using multiple examples. During annotation, they had access to parsed versions of the sentences in question (see Section 5 for details), and to the English FrameNet and SALSA.

The annotation proceeded in three phases: a training phase (40 sentences), a calibration phase (100 sentences), and a production mode phase (1000 sentences). In the calibration phase, sentences were doubly annotated to assess inter-annotator agreement. In production mode, sentences were split into two distinct sets, each of which was annotated by a single coder. We ensured that no annotator saw both parts of any sentence pair to guarantee independent annotation of the bilingual data. Each coder annotated approximately the same amount of data in English and German.

Table 2 shows the results of our inter-annotator agreement study. In addition to the widely used Kappa statistic, we computed a number of different agreement measures: the ratio of frames common between two sentences (Frame Match), the ratio of common roles (Role Match), and the ratio of roles with identical spans (Span Match). As can be seen, annotators tend to agree in frame assignment; disagreements are mainly due to fuzzy distinctions between frames (e.g., between AWARENESS and CERTAINTY). As can be seen from Table 2, annotators agree in what roles to assign (Role Match is 0.95 for both English and German); agreeing on their exact spans is a harder problem.

Measure	English	German	All
Frame Match	0.90	0.87	0.88
Role Match	0.95	0.95	0.95
Span Match	0.85	0.83	0.84
Kappa	0.86	0.90	0.87

Table 2: Monolingual inter-annotation agreement on the calibration set

Measure	Precision	Recall	F-score
Frame Match	0.72	0.72	0.72
Role Match	0.91	0.92	0.91

Table 3: Cross-lingual semantic parallelism between English and German

Semantic Parallelism. Since we obtained parallel FrameNet annotations for English and German, we were able to investigate the degree of semantic parallelism between the two languages. More specifically, we treated the German annotation as gold standard against which we compared the English annotations. To facilitate comparisons with the output of our automatic projection methods (see Section 4), we measured parallelism using precision and recall. Frames and frame roles were counted as matching if they were annotated in a sentence, regardless of their spans. The results are shown in Table 3.

The cross-lingual data exhibit more than twice the amount of frame differences than monolingual data (compare Tables 2 and 3). This indicates that frame disambiguation methods must be employed in automatic role projection to ensure that two aligned tokens evoke the same frame. However, frame disambiguation is outside the scope of the present paper.

On the positive side, role agreement is relatively high (0.91 F-score). This indicates that in cases where frames match across languages, semantic roles could be accurately transferred (provided that these languages diverge little in their argument structure). This observation offers support for the projection approach put forward in this paper. Note, however, that a practical projection system could attain this level of performance only if it could employ an oracle to recover annotators' decisions about the span of roles. We can obtain a more realistic upper bound for an automatic system from the monolingual Role Span agreement figure (F-score 0.84). The latter represents a ceiling for the agreement we can expect from sentences annotated by different annotators.

4 Projection of Semantic Information

In this section, we formalise the semantic projection task and give the details of our modelling approach. All models discussed here project semantic annotations from a source language to a target language. As explained earlier, our present study is only concerned with the projection of roles between matching frames.

4.1 Problem Formulation

We assume that we are provided with source and target sentences represented as sets of entities $e_s \in E_s$ and $e_t \in E_t$. These entities can be words, constituents, phrases, or other groupings. In addition, we are given the semantic annotation of the source sentences from which we can directly read off the source semantic role assignment $a_s : R \rightarrow 2^{E_s}$, where R is the set of semantic roles. The goal of the projection is to specify the target semantic role assignments $a_t : R \rightarrow 2^{E_t}$, which are unknown.[3]

Clearly, effecting the projection requires establishing some form of match between the source and target entities. We therefore formalise projection as a function which maps the source role assignment and a set of matches $M \subseteq E_s \times E_t$ onto a new target role assignment:

$$proj : (A_s \times M) \rightarrow (R \rightarrow 2^{E_t}) \qquad (1)$$

By way of currying, we can state the new target role assignment as a function which directly computes a set of target entities, given the source role assignment, a set of entity matches, and a role:

$$a_t : (A_s \times M \times R) \rightarrow 2^{E_t} \qquad (2)$$

According to this formalisation, the crucial part of semantic projection is to identify a correct and exhaustive set of entity matches. Obviously, this raises the question of what linguistic information is appropriate for establishing M. Unfortunately, any attempt to compute a match based on categorical data derived from linguistic analyses (e.g., parts of speech, phrase types or grammatical relations), needs to empirically derive cross-linguistic similarities between categories, a task which must be repeated for every new language pair, and requires additional data.

$r \in R$	Semantic role
$t_s \in T_s, t_t \in T_t$	Source, target tokens
$al \in Al : T_s \rightarrow 2^{T_t}$	Word alignment
$a_s \in A_s : R \rightarrow 2^{T_s}$	Source role assignment
$a_t : (A_s \times Al \times R) \rightarrow 2^{T_t}$	Projected target role assignment

Table 4: Notation and signature summary for word-based projection

Rather than postulating an ad hoc similarity function, we use word alignments to derive information about semantic roles in the target language. Our first model family (Section 4.2) relies exclusively on this knowledge source. Although potentially useful as a proxy for semantic equivalence, automatically induced alignments are often noisy, thus leading to errors in annotation projection (Yarowsky et al., 2001). For example, function words commonly diverge across languages and are systematically misaligned; furthermore, alignments are restricted to single words rather than word combinations. This observation motivates a second model family with a bias towards linguistically meaningful entities (Section 4.3). Such entities can be constituents derived from the output of a parser or non-recursive syntactic structures (i.e., chunks).

In this paper we compare simple word alignment models against more resource intensive models that utilise constituent-based information and examine whether syntactic knowledge significantly contributes to semantic projection.

4.2 Word-based Projection Model

The first model family uses source and target *word tokens* as entities for projection. In this framework, projection models can be defined by deriving the set of matches M directly from word alignments. The resulting signatures are shown in Table 4.

Our first projection model assigns to each role r with source span $s(r)$ the set of all target tokens which are aligned to a token in the source span:

$$a_w(a_s, al, r) = \bigcup_{t_s \in a_s(r)} al(t_s) \qquad (3)$$

[3]Without loss of generality, we limit ourselves to one frame per sentence, as does FrameNet.

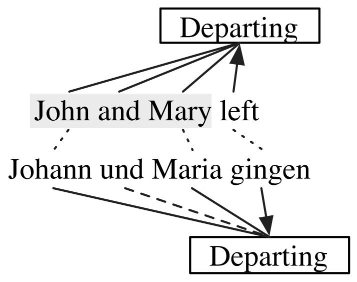


Figure 1: Word alignment-based semantic projection of Role THEME (shadowed), Frame DEPARTING

The main shortcoming of this model is that it cannot capture an important linguistic property of semantic roles, namely that they almost always cover contiguous stretches of text. We can repair non-contiguous projections by applying a "convex complementing" heuristic to the output of (3), which fills all holes in a sequence of tokens, without explicit recourse to syntactic information. We define the convex complementing heuristic as:

$$a_{cw}(a_s, al, r) = \{t_t \mid \min(i(a_{t_1})) \leq i(t_t) \leq \max(i(a_{t_1}))\} \quad (4)$$

where i returns the index of a token t.

The two models just described are illustrated in Figure 1. The frame DEPARTING is introduced by *left* and *gingen* in English and German, respectively. For simplicity, we only show the edges corresponding to the THEME role. In English, the THEME is realised by the words *John and Mary*. The dotted lines show the available word alignments. The projection of the THEME role according to (3) consists only of the tokens {*Johann*, *Maria*} (shown by the plain black lines); the convex complementing heuristic in model (4) adds the token *und*, resulting in the (correct) convex set {*Johann*, *und*, *Maria*}.

4.3 Constituent-based Projection Model

Our second model family attempts to make up for errors in the word alignment by projecting from and to *constituents*. In this study, our constituents are obtained from full parse trees (see Section 5 for details). Models which use non-recursive structures are also possible; however, we leave this to future work.

The main difference from word-based projection models is the introduction of constituent information as an intermediate level; we thus construct a *constituent alignment* for which only a subset of word alignments has to be accurate. The appropriate signatures and notation for constituent-based projection are summarised in Table 5.

In order to keep the model as flexible as possible, and to explore the influence of different design decisions, we model constituent-based projection as two independently parameterisable subtasks: first we compute a real-valued *similarity function* between source and target constituents; then, we employ the similarity function to align relevant constituents and project the role information.

Similarity functions. In principle, any function which matches the signature in Table 5 could be used. In practice, the use of linguistic knowledge runs into the problem of defining similarity between category-based representations discussed above. For this reason, we limit ourselves to two simple similarity functions based on word overlap: Given source and target constituents c_s and c_t, we define the *word overlap* o_w of c_s with c_t as the proportion of tokens within c_t aligned to tokens within c_s. Let $yield(c)$ denote the set of tokens in the yield of a constituent c, then:

$$o_w(c_s, c_t) = \frac{|(\bigcup_{t_s \in yield(c_s)} al(t_s)) \cap yield(c_t)|}{|yield(c_t)|} \quad (5)$$

Since the asymmetry of this overlap measure leads to high overlap scores for small target constituents, we define *word overlap similarity*, as the product of two constituents' mutual overlap:

$$sim(c_s, c_t) = o(c_s, c_t) \cdot o(c_t, c_s) \quad (6)$$

Simple word-based overlap has one undesired characteristic: larger constituents tend to be less similar because of missing alignments (e.g., between function words). Since content words are arguably more important for the role projection task, we define a second overlap measure, *content word overlap* o_{wc}, which takes only nouns, verbs and adjectives into account. Let $yield_c(c)$ denote the set of tokens in the yield of c that are content words, then:

$$o_{wc}(c_s, c_t) = \frac{|(\bigcup_{t_s \in yield_c(c_s)} al(t_s)) \cap yield_c(c_t)|}{|yield_c(c_t)|} \quad (7)$$

Constituent alignment. Considerable latitude is available in interpreting a similarity function to derive a constituent alignment. Due to space limitations, we demonstrate two basic models.

Our first *forward constituent alignment* model (a_{fc}), aligns source constituents that form the span

$r \in R$	Semantic role
$c_s \in C_s, c_t \in C_t$	Source and target constituents
$yield : C \rightarrow T$	Yield of a constituent
$yield_c : C \rightarrow T$	Content word yield of a constituent
$al \in Al : T_s \rightarrow 2^{T_t}$	Word alignment
$a_s \in A_s : R \rightarrow 2^{C_s}$	Source role assignment
$sim : C_s \times C_t \rightarrow \mathbb{R}^+$	Constituent similarity
$a_t : A_s \times Sim \times R \rightarrow 2^{C_t}$	Projected target role assignment

Table 5: Notation and signature summary for constituent-based projection

of a role to a single target constituent. We compute the similarity of a target constituent c_t to a set of source constituents $c_s \in a_s(r)$ by taking the product similarity for each source and target constituent pair:

$$a_{fc}(a_s, sim, r) = \underset{c_t \in C_t}{\operatorname{argmax}} \prod_{c_s \in a_s(r)} sim(c_s, c_t) \quad (8)$$

This projection model forces the target role assignment to be a function, i.e., it makes the somewhat simplifying assumption that each role corresponds to a single target constituent.

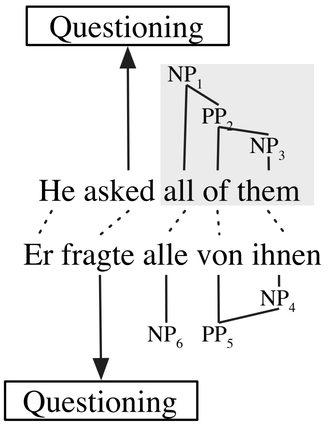
Our second *backward constituent alignment* model (a_{bc}) proceeds in the opposite direction: it iterates over target constituents and attempts to determine their most similar source constituent for each c_t. If the aligned source constituent is labelled with a role, it is projected onto c_t:

$$a_{bc}(a_s, sim, r) = \{c_t \,|\, (\underset{c_s \in C_s}{\operatorname{argmax}}\, sim(c_s, c_t)) \in a_s(r)\} \quad (9)$$

In general, a_{bc} allows for more flexible role projection: it will sometimes decide not to project a role at all (if the source constituents are dissimilar to any target constituents), or it can assign a role to more than one target constituent; however, this means that there is less control over what is projected, and wrong alignments can lead to wrong results more easily.

Finally, if no word alignments are found for complete source or target constituents, the maximal similarity rating in a_{bc} or a_{bf} will be zero. This is often the case for semantically weak single-word constituents such as demonstrative pronouns (e.g., *[That] is right./ [Das] ist richtig.*). When we observe this phenomenon, we heuristically skip unaligned constituents (*zero skipping*).

Figure 2 contrasts the two constituent-based projection models using the frame QUESTIONING as



	NP_1	PP_2	NP_3
NP_4	0.33	0.5	1
PP_5	**0.67**	1	0.5
NP_6	<u>0.33</u>	0	0

Figure 2: Constituent-based semantic projection of role ADDRESSEE (shadowed), frame QUESTIONING. Below: Constituent similarity matrix.

an example. Again, we only show one role, ADDRESSEE, indicated by the shadowed box in Figure 2. Note that the object NP in German was misparsed as an NP and a PP, a relatively frequent error. The difference between the two decision procedures can be explained straightforwardly by looking at the table below the graph, which shows the similarity matrix for the constituents according to equation (6). In this table, the source constituents (indices 1–3) correspond to columns, and the target constituents (indices 4–6) to rows. The alignment model in (8) iterates over labelled source constituents (here only NP_1) and chooses the row with the highest value as the target constituent for a candidate role. In our case, this is the PP_5 (cell in boldface). In contrast, model (9) iterates over all target constituents (i.e., rows) and checks if the most similar source constituent bears a role label. Since NP_1 is the most similar constituent for NP_6 (underlined cell), (9) assigns the QUESTIONING role to NP_6.

5 Experiments

Evaluation Framework. We implemented the models described in the previous section and used them to project semantic information from English onto German. For the constituent-based models, constituent information was obtained from the output of Collins' parser (1997) for English and Dubey's parser (2004) for German. Words were

Model	Precision	Recall	F-score
w	0.41	0.40	0.41
cw	0.46	0.45	0.46
Upper bound	0.85	0.84	0.84

Table 6: Results for word-based projection models

aligned using the default setting[4] of GIZA++ (Och and Ney, 2003), a publicly available implementation of the IBM models and HMM word alignment models. We evaluated the projected roles against the "gold standard" roles obtained from the manual annotation (see Section 3). We also compared our results to the upper bound given by the inter-annotator agreement on the calibration data set.

Results. Table 6 shows our results for the word-based projection models. The simplest word-based model (a_w), obtains an F-score of 0.41. This is a good result considering that the model does not exploit any linguistic information (e.g., parts of speech or syntactic structure). It also supports our hypothesis that word alignments are useful for the role projection task. The convex complementing heuristic (a_{cw}) delivers an F-score increase of five points over the "words only" model, simply by making up for holes in the word alignment.

We evaluated eight instantiations of the constituent-based projection models; the results are shown in Table 7. The best model (in boldface) uses forward constituent alignment, content word-based overlap similarity, and zero skipping. We observe that backward constituent alignment-based models (1–4) perform similarly to word-based projection models (the F-score ranges between 0.40 and 0.45). However, they obtain considerably higher precision (albeit lower recall) than the word-based models. This may be an advantage if the projected data is destined for training target-language semantic parsers. This precision/recall pattern appears to be a direct result of a_{bc}, which only projects a role from c_s to c_t if c_s "wins" against all other source constituents, thus resulting in reliable, but overly cautious projections, which cannot not be further improved by zero skipping.

The forward constituent alignment models (5–8) show consistently higher performance than word-based models and models 1–4, indicating that the stronger assumptions made by forward alignment

[4]The training scheme involved five iterations of Model 1, five iterations of the HMM model, five iterations of Model 3, and five iterations of Model 4.

Model	*al*	*o*	0-skip	Precision	Recall	F-score
1	*bc*	*w*	no	0.70	0.33	0.45
2	*bc*	*w*	yes	0.70	0.33	0.45
3	*bc*	*wc*	no	0.65	0.32	0.42
4	*bc*	*wc*	yes	0.65	0.32	0.42
5	*fc*	*w*	no	0.61	0.60	0.60
6	*fc*	*w*	yes	0.66	0.60	0.63
7	*fc*	*wc*	no	0.62	0.60	0.61
8	*fc*	*wc*	**yes**	**0.70**	**0.60**	**0.65**
Upper bound				0.85	0.84	0.84

Table 7: Results for constituent-based projection models (*al*: constituent alignment model; *o*: overlap measure; 0-skip: zero skipping)

are justified in the data. In addition, we also find that we can increase precision by concentrating on reliable alignments. This is achieved by using the zero skipping heuristic (compare the odd vs. even-numbered models in Table 7) and by computing overlap on content words (compare Models 6 vs. 8, and 5 vs. 7).

We used the χ^2 test to examine whether the differences observed between the two classes of models are statistically significant. The best constituent-based model significantly outperforms the best word-based model both in terms of precision ($\chi^2 = 114.47$, $p < 0.001$) and recall ($\chi^2 = 400.40$, $p < 0.001$). Both projection models perform significantly worse than humans ($p < 0.001$).

Discussion. Our results confirm that constituent information is important for the semantic projection task. Our best model adopts a conservative strategy which enforces a one-to-one correspondence between roles and target constituents. This strategy leads to high precision, however recall lags behind (see Model 8 in Table 7). Manual inspection of the projection output revealed that an important source of missing roles are word alignments gaps. Such gaps are not only due to noisy alignments, but also reflect genuine structural differences between translated sentences. Consider the following (simplified) example for the STATEMENT frame (introduced by *say*) and its semantic role STATEMENT (introduced by *we*):

(10) We claim X and we say Y
Wir behaupten X und — sagen Y

The word alignment correctly aligns the German pronoun *wir* with the first English *we* and leaves the second occurrence unaligned. Since there is no corresponding German word for the second *we*, projection of the SPEAKER role fails. In future work, this problem could be handled with explicit identification of empty categories (see Dienes and Dubey, 2003).

6 Conclusions

In this paper, we argue that parallel corpora show promise in relieving the lexical acquisition bottleneck for low density languages. We proposed semantic projection as a means of obtaining FrameNet annotations automatically without additional human effort. We examined semantic parallelism, a prerequisite for accurate projection, and showed that semantic roles can be successfully projected for predicate pairs with matching frame assignments. Similarly to previous work (Hwa et al., 2002), we find that some mileage can be gained by assuming direct correspondence between two languages. However, linguistic knowledge is key in obtaining meaningful projections. Our experiments show that the use of constituent information yields substantial improvements over relying on word alignment alone. Nevertheless, the word-based models offer a good starting point for low-density languages for which parsers are not available. Their output could be further post-processed manually or automatically using bootstrapping techniques (Riloff and Jones, 1999).

We have presented a general, flexible framework for semantic projection which can be easily applied to other languages. An important direction for future work lies in the assessment of more shallow syntactic information (i.e., chunks) which can be obtained more easily for new languages, and generally in the integration of more linguistic knowledge to guide projection. Finally, we will incorporate into our projection approach automatic semantic role annotations for the source language and investigate the potential of the projected annotations for training semantic parsers for the target language.

Acknowledgements. The authors acknowledge the support of DFG (Padó; grant PI-154/9-2) and EPSRC (Lapata; grant GR/T04540/01). Thanks to B. Kouchnir and P. Kreischer for their annotation.

References

H. C. Boas. 2002. Bilingual framenet dictionaries for machine translation. In *Proceedings of LREC 2002*, 1364–1371, Las Palmas, Canary Islands.

L. Burnard, 1995. *The Users Reference Guide for the British National Corpus*. British National Corpus Consortium, Oxford University Computing Service, 1995.

X. Carreras, L. Màrquez, eds. 2005. *Proceedings of the CoNLL shared task: Semantic role labelling*, 2005.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL/EACL 1997*, 16–23, Madrid, Spain.

P. Dienes, A. Dubey. 2003. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of EMNLP 2003*, 33–40, Sapporo, Japan.

A. Dubey. 2004. *Statistical parsing for German: Modelling syntactic properties and annotation differences.* Ph.D. thesis, Saarland University.

K. Erk, A. Kowalski, S. Padó, M. Pinkal. 2003. Towards a resource for lexical semantics: A large German corpus with extensive semantic annotation. In *Proceedings of ACL 2003*, 537–544, Sapporo, Japan.

C. J. Fillmore, C. R. Johnson, M. R. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

P. Fung, B. Chen. 2004. BiFrameNet: Bilingual frame semantics resources construction by cross-lingual induction. In *Proceedings of COLING 2004*, 931–935, Geneva, Switzerland.

D. Gildea, D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

R. Hwa, P. Resnik, A. Weinberg, O. Kolak. 2002. Evaluation translational correspondance using annotation projection. In *Proceedings of ACL 2002*, 392–399, Philadelphia, PA.

P. Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation. Draft.

S. Narayanan, S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of COLING 2004*, 693–701, Geneva, Switzerland.

F. J. Och, H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

S. Padó, M. Lapata. 2005. Cross-lingual bootstrapping for semantic lexicons. In *Proceedings of AAAI 2005*, Pittsburgh, PA.

E. Riloff, R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI 1999*, Orlando, FL.

D. A. Smith, N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP 2004*, 49–56, Barcelona, Spain.

M. Surdeanu, S. Harabagiu, J. Williams, P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*, 8–15, Sapporo, Japan.

D. Yarowsky, G. Ngai, R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT 2001*, 161–168.

OCR Post-Processing for Low Density Languages

Okan Kolak
Computer Science and UMIACS
University of Maryland
College Park, MD 20742
okan@umiacs.umd.edu

Philip Resnik
Linguistics and UMIACS
University of Maryland
College Park, MD 20742
resnik@umiacs.umd.edu

Abstract

We present a lexicon-free post-processing method for optical character recognition (OCR), implemented using weighted finite state machines. We evaluate the technique in a number of scenarios relevant for natural language processing, including creation of new OCR capabilities for low density languages, improvement of OCR performance for a native commercial system, acquisition of knowledge from a foreign-language dictionary, creation of a parallel text, and machine translation from OCR output.

1 Introduction

The importance of rapidly retargeting existing natural language processing (NLP) technologies to new languages is widely accepted (Oard, 2003). Statistical NLP models have a distinct advantage over rule based approaches to achieve this goal, as they require far less manual labor; however, training statistical NLP methods requires on-line text, which can be hard to find for so-called “low density” languages — that is, languages where few on-line resources exist. In addition, for many languages of interest input data are available mostly in printed form, and must be converted to electronic form prior to processing.

Optical character recognition (OCR) is often the only feasible method to perform this conversion, owing to its speed and cost-effectiveness. Unfortunately, the performance of OCR systems is far from perfect and recognition errors significantly degrade the performance of NLP applications. This is true both in resource acquisition, such as automated bilingual lexicon generation (Kolak et al., 2003), and for end-user applications such as rapid machine translation (MT) in the battlefield for document filtering (Voss and Ess-Dykema, 2000). Moreover, for low density languages, there simply may not be an OCR system available.

In this paper, we demonstrate that via statistical post-processing of existing systems, it is possible to achieve reasonable recognition accuracy for low density languages altogether lacking an OCR system, to significantly improve on the performance of a trainable commercial OCR system, and even to improve significantly on a native commercial OCR system.[1] By taking a post-processing approach, we require minimal assumptions about the OCR system used as a starting point.

The proper role of our post-processing approach depends on the language. For languages with little commercial potential for OCR, it may well provide the most practical path for language-specific OCR development, given the expensive and time consuming nature of OCR development for new languages and the “black box” nature of virtually all state-of-the-art OCR systems. For languages where native OCR development may take place, it is a fast, practical method that allows entry into a new language until native OCR development catches up. For these, and also for languages where native systems exist,

[1]Currently we assume the availability of an OCR system that supports the script of the language-of-interest, or which is script independent (Natarajan et al., 2001).

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 867–874, Vancouver, October 2005. ©2005 Association for Computational Linguistics

we show that post-processing can yield improvements in performance.

Sections 2 and 3 describe the method and its implementation. In Section 4 we cover a variety of relevant NLP scenarios: Creating OCR capabilities for Igbo, performing OCR on a dictionary for Cebuano, using OCR to acquire the Arabic side of a common parallel text, and evaluating the value of OCR post-processing for machine translation of Arabic and Spanish. In Sections 5 and 6 we discuss related work and summarize our findings.

2 Post-Processing System

We use the noisy channel framework to formulate the correction problem, revising our previous model (Kolak et al., 2003). That model takes the form

$$P(O,b,a,C,W) = P(O,b|a,C,W)P(a|C,W)P(C|W)P(W)$$

whose components are a word-level source model $P(W)$, a word-to-character model $P(C|W)$, a segmentation model $P(a|C,W)$, and a model for character sequence transformation, $P(O,b|a,C,W)$. W is the correct word sequence and C is the corresponding character sequence, which is recognized as O by the OCR system. a and b are segmentation vectors for C and O.

The original model requires a lexicon that covers all words in the processed text — a strong assumption, especially for low density languages. We converted the model into a character-based one, removing the need for a lexicon. Generation of W is replaced by generation of C, which renders $P(C|W)$ irrelevant, and the model becomes

$$P(O,b,a,C) = P(O,b|a,C)P(a|C)P(C)$$

Although word-based models generally perform better, moving from words to characters is a necessary compromise because word-based models are useless in the absence of a lexicon, which is the case for many low-density languages.

In addition to eliminating the need for a lexicon, we developed a novel method for handling word merge/split errors.[2] Rather than modeling these errors explicitly using a segmentation model, we simply treat them as character deletion/insertion errors involving the space character, allowing us to handle them within the error model. The segmentation step is absorbed into the character transformation step, so a and b are no longer necessary, hence the final equation becomes

$$P(O,C) = P(O|C)P(C)$$

which is a direct application of the noisy channel model. We can describe the new generative process as follows: First, a sequence of characters C are generated, with probability $P(C)$, and the OCR system converts it into O with probability $P(O|C)$. For example, if the actual input was a_car and it was recognized as $ajar$, $P(ajar, a\_car) = P(ajar|a\_car)P(a\_car)$. Using the channel model to address word merge/split errors without actually using a word level model is, to our knowledge, a novel contribution of our approach.

3 Implementation

We implemented our post-processing system using the framework of weighted finite state machines (WFSM), which provides a strong theoretical foundation and reduces implementation time, thanks to freely available toolkits, such as the AT&T FSM Toolkit (Mohri et al., 1998). It also allows easy integration of our post-processor with numerous NLP applications that are implemented using FSMs (e.g. (Knight and Graehl, 1997; Kumar and Byrne, 2003)).

3.1 Source Model

The source model assigns probability $P(C)$ to original character sequences, C. We use character level n-gram language models as the source model, since n-gram models are simple, easy to train, and usually achieve good performance. More complicated models that make use of constraints imposed by a particular language, such as vowel harmony, can be utilized if desired. We used the CMU-Cambridge Language Modeling Toolkit v2 (Clarkson and Rosenfeld, 1997) for training, using Witten-Bell smoothing and vocabulary type 1; all other parameters were left at their default values.

[2] A merge error occurs when two or more adjacent items are recognized as one, and a split error occurs when an item is recognized as two or more items. These errors can happen both at word level and character level.

3.2 Channel Model

The channel model assigns a probability to O given that it was generated from C. We experimented with two probabilistic string edit distance models for implementing the channel model. The first, following our earlier model (2003), permits single-character substitutions, insertions, and deletions, with associated probabilities. For example, $P(ajar|a\textvisiblespace car) \approx P(a{\mapsto}a)P(\textvisiblespace{\mapsto}\epsilon)P(c{\mapsto}j)P(a{\mapsto}a)P(r{\mapsto}r)$. Note that we are only considering the most likely edit sequence here, as opposed to summing over all possible ways to convert $a\textvisiblespace car$ to $ajar$. The second is a slightly modified version of the spelling correction model of Brill and Moore (2000).[3] This model allows many-to-many edit operations, which makes $P(liter|litre) \approx P(l{\mapsto}l)P(i{\mapsto}i)P(tre{\mapsto}ter)$ possible. We will refer to the these as the single-character (SC) and multi-character (MC) error models, respectively.

We train both error models over a set of corresponding ground truth and OCR sequences, $\langle C, O\rangle$. Training is performed using expectation-maximization: We first find the most likely edit sequence for each training pair to update the edit counts, and then use the updated counts to re-estimate edit probabilities. For MC, after finding the most likely edit sequence, extended versions of each non-copy operation that include neighboring characters are also considered, which allows learning any common multi-character mappings. Following Brill and Moore, MC training performs only one iteration of expectation-maximization.

In order to reduce the time and space requirements of the search at correction time, we impose a limit on number of errors per token. Note that this is not a parameter of the model, but a limit required by its computational complexity. A lower limit will almost always result in lower correction performance, so the highest possible limit allowed by time and memory constraints should be used. It is possible to correct more errors per token by iterating the correction process. However, iterative correction cannot guarantee that the result is optimal under the model.

[3]We ignore the location of the error within the word, since it is not as important for OCR as it is for spelling.

3.3 Chunking

Since we do not require a lexicon, we work on lines of text rather than words. Unfortunately the search space for correcting a complete line is prohibitively large and we need a way to break it down to smaller, independent chunks. The chunking step is not part of the model, but rather a pre-processing step: chunks are identified, each chunk is corrected independently using the model, and the corrected chunks are put back together to generate the output.

Spaces provide a natural break point for chunks. However, split errors complicate the process: if parts of a split word are placed in different chunks, the error cannot be corrected. For example, in Figure 1, chunking (b) allows the model to produce the desired output, but chunking (a) simply does not allow combining “sam” and “ple” into “sample”, as each chunk is corrected independently.

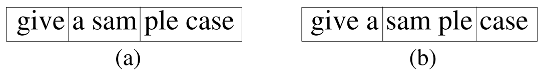


Figure 1: Example of a bad and a good chunking

We address this by using the probabilities assigned to spaces by the source model for chunking. We break the line into two chunks using the space with the highest probability and repeat the process recursively until all chunks are reduced to a reasonable size, as defined by time and memory limitations. Crucially, spurious spaces that cause split errors are expected to have a low probability, and therefore breaking the line using high probability spaces reduces the likelihood of placing parts of a split word in different chunks.

If a lexicon does happen to be available, we can use it to achieve more reliable chunking, as follows. The tokens of the input line that are present in the lexicon are assumed to be correct. We identify runs of out-of-lexicon tokens and attempt to correct them together, allowing us to handle split errors. Note that in this case the lexicon is used only to improve chunking, not for correction. Consequently, coverage of the lexicon is far less important.

Our lexicon-free chunking algorithm placed an erroneous boundary at 11.3% of word split points for Arabic test data (Section 4.3). However, correction performance was identical to that of error-

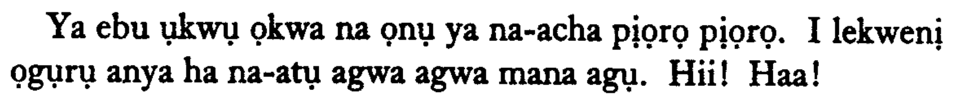

Ya ebu ụkwụ ọkwa na ọnụ ya na-acha pịọrọ pịọrọ. I lekwenị ọgụrụ anya ha na-atụ agwa agwa mana agụ. Hii! Haa!

Figure 2: A small excerpt from Akụkọ

free chunking.[4] Incorrect decisions did not hurt because the correction method was not able to fix those particular split errors, regardless. The errors of the chunking and correction models coincided as they both rely on the same language model. Therefore, chunking errors are unlikely to reduce the correction performance.

3.4 Correction

Correction is performed by estimating the most probable source character sequence $\hat{C}$ for a given observed character sequence O, using the formula:

$$\hat{C} = \underset{C}{\operatorname{argmax}}\{P(O|C)P(C)\}$$

We first encode O as an FSA and compose it with the inverted error model FST.[5] The resulting FST is then composed with the language model FSA. The final result is a lattice that encodes all feasible sequences C, along with their probabilities, that could have generated O. We take the sequence associated with the most likely path through the lattice as $\hat{C}$.

4 Evaluation

We evaluate our work on OCR post-processing in a number of scenarios relevant for NLP, including creation of new OCR capabilities for low density languages, improvement of OCR performance for a native commercial system, acquisition of knowledge from a foreign-language dictionary, creation of a parallel text, and machine translation from OCR output. The languages studied include Igbo, Cebuano, Arabic, and Spanish.

For intrinsic evaluation, we use the conventional Word Error Rate (WER) metric, which is defined as

$$WER(C, O) = \frac{WordEditDistance(C, O)}{WordCount(C)}$$

We do not use the Character Error Rate (CER) metric, since for almost all NLP applications the unit of information is the words. For extrinsic evaluation of machine translation, we use the BLEU metric (Papineni et al., 2002).

4.1 Igbo: Creating an OCR System

Igbo is an African language spoken mainly in Nigeria by an estimated 10 to 18 million people, written in Latin script. Although some Igbo texts use diacritics to mark tones, they are not part of the official orthography and they are absent in most printed materials. Other than grammar books, texts for Igbo, even hardcopy, are extremely difficult to obtain. To our knowledge, the work reported here creates the first OCR system for this language.

For the Igbo experiments, we used two sources. The first is a small excerpt containing 6727 words from the novel "Juo Obinna" (Ubesie, 1993). The second is a small collection of short stories named "Akụkọ Ife Nke Ndị Igbo" (Green and Onwuamaegbu, 1970) containing 3544 words. We will refer to the former as "Juo" and the latter as "Akụkọ" hereafter. We generated the OCR data using a commercial English OCR system.[6] Juo image files were generated by scanning 600dpi laser printer output at 300dpi resolution. Akụkọ image files were generated by scanning photocopies from the bound hardcopy at 300dpi. Figure 2 provides a small excerpt from the actual Akụkọ page images used for recognition. For both texts, we used the first two thirds for training and the remaining third for testing.

We trained error and language models (EMs and LMs) using the training sets for Juo and Akụkọ separately, and performed corrections of English OCR output using different combinations of these models on both test sets. Table 1 shows the results for the Juo test set while Table 2 presents the results for Akụkọ. The relative error reduction ranges from 30% to almost 80%. The SC error model performs better than the MC error model under all conditions.

[4]Ignoring errors that result in valid words, lexicon-based chunking is always error-free.

[5]Inversion reverses the direction of the error model, mapping observed sequences to possible ground truth sequences.

[6]Abby Fine Reader Professional Edition Version 7.0

Conditions			Results	
LM Data	EM Data	EM type	WER (%)	Red. (%)
Juo	Juo	MC	8.66	74.18
Juo	Akụkọ	MC	15.23	54.59
Akụkọ	Juo	MC	13.25	60.49
Akụkọ	Akụkọ	MC	19.08	43.11
Juo	**Juo**	**SC**	**7.11**	**78.80**
Juo	Akụkọ	SC	11.49	65.74
Akụkọ	Juo	SC	13.42	59.99
Akụkọ	Akụkọ	SC	18.92	43.59
Original OCR Output			**35.44**	-

Table 1: Post-correction WER for English OCR on Juo

Conditions			Results	
LM Data	EM Data	EM type	WER (%)	Red. (%)
Juo	Juo	MC	21.42	36.33
Juo	Akụkọ	MC	18.08	46.25
Akụkọ	Juo	MC	21.51	36.06
Akụkọ	Akụkọ	MC	18.16	46.02
Juo	Juo	SC	19.92	40.78
Juo	Akụkọ	SC	16.49	50.98
Akụkọ	Juo	SC	19.92	40.78
Akụkọ	**Akụkọ**	**SC**	**16.40**	**51.25**
Original OCR Output			**33.64**	-

Table 2: Post-correction WER for English OCR on Akụkọ

This is due to the fact that MC requires more training data than SC. Furthermore, most of the errors in the data did not require many-to-many operations. Results in Tables 1 and 2 are for 6-gram language model and error limit of 5; corresponding 3-gram error rates were 1% to 2% (absolute) higher.

The best correction performance is achieved when both the EM and LM training data come from the same source as the test data, almost doubling the performance achieved when they were from a different source.[7] Note that the amount of training data is small, four to eight pages, so optimizing performance via manual entry of document-specific training text is not unrealistic for scenarios involving long documents such as books.

4.1.1 Using a Trainable OCR System

In an additional experiment with Igbo, we found that post-processing can improve performance substantially even when an OCR system trained on Igbo characters is the starting point. In particular, the commercial OCR system used for Igbo experiments supports user-trained character shape models. Using Juo as the source, we trained the commercial OCR system manually on Igbo characters, resulting in a 7.43% WER on Juo without postprocessing.[8] Note that this is slightly higher than the 7.11% WER achieved using an English OCR system together with our post-processing model. We used a 6-gram LM, and a SC EM with error limit of 5. Table 3 shows that by post-processing the Igbo-trained OCR system, we reduce the word error rate by 50%.

Conditions		Results	
LM Data	EM Data	WER (%)	Red. (%)
Juo	**Juo**	**3.69**	**50.34**
Juo	Akụkọ	5.24	29.48
Akụkọ	Juo	5.08	31.63
Akụkọ	Akụkọ	7.38	0.67
Original OCR Output		**7.43**	-

Table 3: Post-correction WER for trained OCR system on Juo

4.2 Cebuano: Acquiring a Dictionary

Cebuano is a language spoken by about 15 million people in the Philippines, written in Latin script. The scenario for this experiment is converting a Cebuano hardcopy dictionary into electronic form, as in DARPA's Surprise Language Dry Run (Oard, 2003). The dictionary that we used had diacritics, probably to aid in pronunciation. The starting-point OCR data was generated using a commercial OCR system.[9] The fact that the tokens to be corrected come from a dictionary means (1) there is little context available and (2) word usage frequencies are not reflected. Character-based models may be affected by these considerations, but probably not to the extent that word-based models would be.

Table 4 shows WER for Cebuano after post-processing. The *size* column represents the number of dictionary entries used for training, where each entry consists of one or more Cebuano words. As can be seen from the table, our model reduces WER substantially for all cases, ranging from 20% to 50% relative reduction. As expected, the correction performance increases with the amount of training data; note, however, that we achieve reasonable correction performance even using only 500 dictionary entries for training.

[7]There was no overlap between training and test data under any circumstance.

[8]The system trains by attempting OCR on a document and asking for the correct character whenever it is not confident.

[9]ScanSoft Developer's Kit 2000, which has no built-in support for Cebuano.

Conditions			Results	
Size	LM	EM	WER (%)	Red. (%)
500	3-gram	SC	5.37	33.04
500	3-gram	MC	5.05	37.03
500	6-gram	SC	6.41	20.07
500	6-gram	MC	5.33	33.54
1000	3-gram	SC	5.33	33.54
1000	3-gram	MC	4.63	42.27
1000	6-gram	SC	5.58	30.42
1000	6-gram	MC	4.67	41.77
27363	3-gram	SC	4.34	45.89
27363	3-gram	MC	4.14	48.38
27363	6-gram	SC	4.55	43.27
27363	**6-gram**	**MC**	**3.97**	**50.50**
Original OCR Output			**8.02**	-

Table 4: Post-correction WER for Cebuano

Contrary to the Igbo results, the MC error model performs better than the SC error model. And, interestingly, the 3-gram language model performs better than the 6-gram model, except for the largest training data and MC error model combination. Both differences are most likely caused by the implications of using a dictionary as discussed above.

4.3 Arabic: Acquiring Parallel Text

We used Arabic to illustrate conversion from hardcopy to electronic text for a widely available parallel text, the Bible (Resnik et al., 1999; Kanungo et al., 2005; Oard, 2003). We divided the Bible into ten equal size segments, using the first segment for training the error model, the first nine segments for the language model, and the first 500 verses from the last segment for testing. Since diacritics are only used in religious text, we removed all diacritics. The OCR data was generated using a commercial Arabic OCR system.[10] Note that this evaluation differs from Igbo and Cebuano, as the experiments were performed using an existing *native* OCR system. It also allowed us to evaluate chunking, as Arabic data has far more word merge/split errors compared to Igbo and Cebuano.

Table 5 shows the correction performance for Arabic under various conditions. The *Limit* column lists the maximum number of errors per token allowed and the *M/S* column indicates whether correction of word merge/split errors was allowed. We achieve significant reductions in WER for Arabic. The first two rows show that the 6-gram language model performs much better than the 3-gram model. Interestingly, higher order n-grams perform worse when we allow word merge/split errors. Note that for handling word merge/split errors we need to learn the character distributions within lines, rather than within words as we normally do. Consequently, more training data is required for reliable parameter estimation. Handling word merge/split errors improve the performance, which is expected. Allowing fewer errors per token reduces the performance, since it is not possible to correct words that have more character errors than the limit. Unfortunately, increasing the error limit increases the search space exponentially, making it impossible to use high limits. As mentioned in Section 3.2, iterative correction is a way to address this problem.

[10]Sakhr Automatic Reader Version 6.0

Conditions			Results	
M/S	LM	Limit	WER (%)	Red. (%)
no	3-gram	2	22.14	10.33
no	6-gram	2	17.99	27.14
yes	3-gram	2	18.26	26.04
yes	**3-gram**	**4**	**17.74**	**28.15**
yes	5-gram	2	20.74	16.00
Original OCR Output			**24.69**	-

Table 5: Post-correction WER for Arabic

4.4 Extrinsic Evaluation: MT

While our post-processing methods reduce WER, our main interest is their impact on NLP applications. We have performed machine translation experiments to measure the effects of OCR errors and the post-processing approach on NLP application performance.

For Arabic, we trained a statistical MT system using the first nine sections of the Bible data. The language model is trained using the CMU-Cambridge toolkit and the translation model using the GIZA++ toolkit (Och and Ney, 2000). We used the ReWrite decoder (Germann, 2003) for translation.

BLEU scores for OCR, corrected, and clean text were 0.0116, 0.0141, and 0.0154, respectively. This establishes that OCR errors degrade the performance of the MT system, and we are able to bring the performance much closer to the level of performance on clean text by using post-processing. Clearly the BLEU scores are quite low; we are planning to perform experiments on Arabic using a more advanced translation system, such as Hiero (Chiang, 2005).

MT System	Input Text	BLEU Score
Systran	OCR	0.2000
Systran	Corrected	0.2606
Systran	Clean	0.3188
ReWrite	OCR	0.1792
ReWrite	Corrected	0.2234
ReWrite	Clean	0.2590

Table 6: Spanish-English translation results

In order to test in a scenario with better translation performance, we performed MT evaluations using Spanish. We used a commercial translation system, Systran, in addition to statistical translation. More resources being available for this language, corrected text for Spanish experiments was obtained using our original model that takes advantage of a lexicon (2003). Table 6 shows that scores are much higher compared to Arabic, but the pattern of improvements using post-processing is the same.

5 Related Work

There has been considerable research on automatic error correction in text. Kukich (1992) provides a general survey of the research in the area. Unfortunately, there is no standard evaluation benchmark for OCR correction, and implementations are usually not publicly available, making a direct comparison difficult.

Most correction methods are not suitable for low density languages as they rely on lexicons. Goshtasby and Ehrich (1988) present a lexicon-free method based on probabilistic relaxation labeling. However, they use the probabilities assigned to individual characters by the OCR system, which is not always available. Perez-Cortes et al. (2000) describe a method which does not have this limitation. They use a stochastic FSM that accepts the smallest k-testable language consistent with a representative sample. While the method can handle words not in its lexicon in theory, it was evaluated using a large k to restrict corrections to the lexicon. They report reducing error rate from 33% to below 2% on OCR output of hand-written Spanish names.

In addition to providing alternatives, the literature provides complementary methods. Guyon and Pereira (1995) present a linguistic post-processor based on variable memory length Markov models that is designed to be used as the language model component of character recognizers. Their model can be used as the source model for our method. Since it is a variable length model, it can allow us to handle higher order n-grams.

A script-independent OCR system is presented by Natarajan et al. (2001). The system is evaluated on Arabic, Chinese, and English, achieving 0.5% to 5% CER under various conditions. Since our post-processing method can be used to reduce the error rate of a trained OCR system, the two methods can be combined to better adapt to new languages.

Voss and Ess-Dykema (2000) evaluated the effects of OCR errors on MT in the context of the FALCon project, which combines off-the-shelf OCR and MT components to provide crude translations for filtering. They report significant degradation in translation performance as a result of OCR errors. For instance, for the Spanish system, OCR process reduced the number of words that can be recognized by the translation module by more than 60%.

6 Conclusions

We have presented a statistical post-processing method for OCR error correction that requires minimal resources, aimed particularly at low density languages and NLP scenarios. The technique gains leverage from existing OCR systems, enabling both minimal-labor adaptation of systems to new low density languages and improvements in native OCR performance.

We rigorously evaluated our approach using real OCR data, and have shown that we can achieve recognition accuracy lower than that achieved by a trainable OCR system for a new language. For Igbo, a very low density language, adapting English OCR achieved relative error reductions as high as 78%, resulting in 7.11% WER. We also showed that the error rate of a trainable OCR system after training can be further reduced up to 50% using post-processing, achieving a WER as low as 3.7%. Post-processing experiments using Cebuano validate our approach in a dictionary-acquisition scenario, with a 50.5% relative reduction in error rate from 8.02% to 3.97%. Evaluation on Arabic demonstrated that the error rate for a native commercial OCR system can be reduced by nearly 30%. In addition, we measured the impact of post-processing on machine translation, quantifying OCR degradation of MT performance and showing that our technique moves the performance of MT on OCR data significantly closer to performance on clean input. See Kolak (forthcoming) for more details and discussion.

One limitation of our approach is its reliance on an existing OCR system that supports the script of the language of interest. Trainable OCR systems are the only option if there is no OCR system that supports the script of interest; however, training an OCR system from scratch is usually a tedious and time consuming task. Post-processing can be used to reduce the training time and improve recognition accuracy by aiding generation of more training data once basic recognition capability is in place.

Acknowledgments

This research was supported in part by Department of Defense contract RD-02-5700 and DARPA/ITO Cooperative Agreement N660010028910. We are grateful to Mohri et al. for the AT&T FSM Toolkit and Clarkson and Rosenfeld for the CMU-Cambridge Toolkit. We thank David Doermann and his students for the Cebuano text; Chinedu Uchechukwu for the Igbo text and useful information on the language. We also thank Mustafa Murat Tıkır for his help in generating Igbo ground truth for Akụkọ, and for useful discussion.

References

Eric Brill and Robert C. Moore. 2000. An improved model for noisy channel spelling correction. In *Proceedings of the ACL-00*, Hong Kong, China, October.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL-05*, pages 263–270, Ann Arbor, Michigan, USA, June.

Philip Clarkson and Ronald Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of the ESCA Eurospeech*, Rhodes, Greece.

Ulrich Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proceedings of the HLT-NAACL-03*, Edmonton, Alberta, Canada, May.

Ardeshir Goshtasby and Roger W. Ehrich. 1988. Contextual word recognition using probabilistic relaxation labeling. *Pattern Recognition*, 21(5):455–462.

M. M. Green and M. O. Onwuamaegbu, editors. 1970. *Akụkọ Ife Nke Ndị Igbo*. Oxford University Press, Ibadan, Nigeria.

Isabelle Guyon and Fernando Pereira. 1995. Design of a linguistic postprocessor using variable memory length Markov models. In *Proceedings of the ICDAR-95*, volume 1, Montreal, Quebec, Canada, August.

Tapas Kanungo, Philip Resnik, Song Mao, Doe wan Kim, and Qigong Zheng. 2005. The Bible and multilingual optical character recognition. *Communications of the ACM*, 48(6):124–130.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the ACL-97*, Madrid, Spain, July.

Okan Kolak, William Byrne, and Philip Resnik. 2003. A generative probabilistic OCR model for NLP applications. In *Proceedings of the HLT-NAACL-03*, Edmonton, Alberta, Canada, May.

Okan Kolak. forthcoming. *Cross-Lingual Utilization of NLP Resources for New Languages*. Ph.D. thesis, University of Maryland, College Park, Maryland, USA.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, December.

Shankar Kumar and William Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the HLT-NAACL-03*, Edmonton, Alberta, Canada, May.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 1998. A rational design for a weighted finite-state transducer library. *Lecture Notes in Computer Science*, 1436.

Premkumar Natarajan, Zhidong Lu, Richard Schwartz, Issam Bazzi, and John Makhoul. 2001. Multilingual machine printed ocr. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):43–63.

Douglas W. Oard. 2003. The surprise language exercises. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):79–84, June.

Franz. J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the ACL-00*, pages 440–447, Hongkong, China, October.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the ACL-02*, Philedelphia, Pennsylvania, USA, July.

Juan Carlos Perez-Cortes, Juan-Carlos Amengual, Joaquim Arlandis, and Rafael Llobet. 2000. Stochastic error-correcting parsing for OCR post-processing. In *Proceedings of the ICPR-00*, Barcelona, Spain, September.

Philip Resnik, Mari Broman Olsen, and Mona Diab. 1999. The Bible as a parallel corpus: Annotating the 'Book of 2000 Tongues'. *Computers and the Humanities*, 33(1-2):129–153.

Tony Uchenna Ubesie. 1993. *Juo Obinna*. University Press PLC, Ibadan, Nigeria. ISBN: 19575395X.

Clare R. Voss and Carol Van Ess-Dykema. 2000. When is an embedded MT system 'good enough' for filtering? In *Proceedings of the Workshop on Embedded MT Systems, ANLP-NAACL-00*, Seattle, Washington, USA, May.

Inducing a multilingual dictionary from a parallel multitext in related languages

Dmitriy Genzel
Department of Computer Science
Box 1910
Brown University
Providence, RI 02912, USA
dg@cs.brown.edu

Abstract

Dictionaries and word translation models are used by a variety of systems, especially in machine translation. We build a multilingual dictionary induction system for a family of related resource-poor languages. We assume only the presence of a single medium-length multitext (the Bible). The techniques rely upon lexical and syntactic similarity of languages as well as on the fact that building dictionaries for several pairs of languages provides information about other pairs.

1 Introduction and Motivation

Modern statistical natural language processing techniques require large amounts of human-annotated data to work well. For practical reasons, the required amount of data exists only for a few languages of major interest, either commercial or governmental. As a result, many languages have very little computational research done in them, especially outside the borders of the countries in which these languages are spoken. Some of these languages are, however, major languages with hundreds of millions of speakers. Of the top 10 most spoken languages, Linguistic Data Consortium at University of Pennsylvania, the premier U.S. provider of corpora, offers text corpora only in 7 (The World Factbook (2004), 2000 estimate) Only a few of the other languages (French, Arabic, and Czech) have resources provided by LDC. Many Asian and Eastern European languages number tens of millions of speakers, yet very few of these seem to have any related computational linguistics work, at least as presented at the international conferences, such as the ACL.[1]

The situation is not surprising, nor is it likely to significantly change in the future. Luckily, most of these less-represented languages belong to language families with several prominent members. As a result, some of these languages have siblings with more resources and published research. [2] Interestingly, the better-endowed siblings are not always the ones with more native speakers, since political considerations are often more important.[3] If one is able to use the resources available in one language (henceforth referred to as *source*) to facilitate the creation of tools and resource in another, related language (*target*), this problem would be alleviated. This is the ultimate goal of this project, but in the first stage we focus on multi-language dictionary induction.

Building a high-quality dictionary, or even better, a joint word distribution model over all the languages in a given family is very important, because using such a model one can use a variety of techniques to project information across languages, e.g. to parse or to translate. Building a unified model for more than a pair of languages improves the quality over building several unrelated pairwise models, because relating them to each other provides additional information. If we know that word a in language A has as its likely translation word b in language B, and b is translated as c in C, then we also know that a is likely to be translated as c, without looking at

[1]The search through ACL Anthology, for e.g., Telugu (∼70 million speakers) shows only casual mention of the language.

[2]Telugu's fellow Dravidian language *Tamil* (∼65 million speakers) has seen some papers at the ACL

[3]This is the case with Tamil vs. Telugu.

the A to C model.

2 Previous Work

There has been a lot of work done on building dictionaries, by using a variety of techniques. One good overview is Melamed (2000). There is work on lexicon induction using string distance or other phonetic/orthographic comparison techniques, such as Mann and Yarowsky (2001) or semantic comparison using resources such as WordNet (Kondrak, 2001). Such work, however, primarily focuses on finding cognates, whereas we are interested in translations of all words. Moreover, while some techniques (e.g., Mann and Yarowsky (2001)) use multiple languages, the languages used *have* resources such as dictionaries between some language pairs. We do not require any dictionaries for any language pair.

An important element of our work is focusing on more than a pair of languages. There is an active research area focusing on multi-source translation (e.g., Och and Ney (2001)). Our setting is the reverse: we do not use multiple dictionaries in order to translate, but translate (in a very crude way) in order to build multiple dictionaries.

Many machine translation techniques require dictionary building as a step of the process, and therefore have also attacked this problem. They use a variety of approaches (a good overview is Koehn and Knight (2001)), many of which require advanced tools for both languages which we are not able to use. They also use bilingual (and to some extent monolingual) corpora, which we do have available. They do not, however, focus on related languages, and tend to ignore lexical similarity [4], nor are they able to work on more than a pair of languages at a time.

It is also worth noting that there has been some MT work on related languages which explores language similarity in an opposite way: by using dictionaries and tools for both languages, and assuming that a near word-for-word approach is reasonable (Hajic et al., 2000).

[4]Much of recent MT research focuses on pairs of languages which are not related, such as English-Chinese, English-Arabic, etc.

3 Description of the Problem

Let us assume that we have a group of related languages, $L_1 \ldots L_n$, and a parallel sentence-aligned multitext C, with corresponding portions in each language denoted as $C_1 \ldots C_n$. Such a multitext exists for virtually all the languages in the form of the Bible. Our goal is to create a multilingual dictionary by learning the joint distribution $P(x_1 \ldots x_n)_{x_i \in L_i}$ which is simply the expected frequency of the n-tuple of words in a completely word-aligned multitext. We will approach the problem by learning pairwise language models, although leaving some parameters free, and then combine the models and learn the remaining free parameters to produce the joint model.

Let us, therefore, assume that we have a set of models $\{P(x, y|\theta_{ij})_{x \in L_i, y \in L_j}\}_{i \neq j}$ where θij is a parameter vector for pairwise model for languages L_i and L_j. We would like to learn how to combine these models in an optimal way. To solve this problem, let us first consider a simpler and more general setting.

3.1 Combining Models of Hidden Data

Let X be a random variable with distribution $P_{\text{true}}(x)$, such that no direct observations of it exist. However, we may have some indirect observations of X and have built several models of X's distribution, $\{P_i(x|\theta_i)\}_{i=1}^n$, each parameterized by some parameter vector θ_i. P_i also depends on some other parameters that are fixed. It is important to note that the space of models obtained by varying θ_i is only a small subspace of the probability space. Our goal is to find a good estimate of $P_{\text{true}}(x)$.

The main idea is that if some P_i and P_j are close (by some measure) to P_{true}, they have to be close to each other as well. We will therefore make the assumption that if some models of X are close to each other (and we have reason to believe they are fair approximations of the true distribution) they are also close to the true distribution. Moreover, we would like to set the parameters θ_i in such a way that $P(x_i|\theta_i)$ is as close to the other models as possible. This leads us to look for an estimate that is as close to all of our models as possible, under the optimal values of θ_i's, or more formally:

$$P_{\text{est}} = \arg\min_{\hat{P}(\cdot)} \min_{\theta_1} \ldots \min_{\theta_n} d(\hat{P}(\cdot), P_1(\cdot|\theta_1), \ldots P_n(\cdot|\theta_n))$$

where d measures the distance between $\hat{P}$ and all the P_i under the parameter setting θ_i. Since we have no reason to prefer any of the P_i, we choose the following symmetric form for d:

$$\sum_{i=1}^{n} D(\hat{P}(\cdot)||P_i(\cdot|\theta_i))$$

where D is a reasonable measure of distance between probability distributions. The most appropriate and the most commonly used measure in such cases in the Kullback-Leibler divergence, also known as relative entropy:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

It turns out that it is possible to find the optimal $\hat{P}$ under these circumstances. Taking a partial derivative and solving, we obtain:

$$\hat{P}(x) = \frac{\prod_{i=1}^{n} P_i(x|\theta_i)^{1/n}}{\sum_{x' \in X} \prod_{i=1}^{n} P_i(x'|\theta_i)^{1/n}}$$

Substituting this value into the expression for function d, we obtain the following distance measure between the P_i's:

$$\begin{aligned} & d'(P_1(X|\theta_1) \ldots P_n(X|\theta_n)) \\ &= \min_{\hat{P}} d(\hat{P}, P_1(X|\theta_1), \ldots P_n(X|\theta_n)) \\ &= -\log \textstyle\sum_{x \in X} \prod_{i=1}^{n} P_i(x|\theta_i)^{1/n} \end{aligned}$$

This function is a generalization of the well-known Bhattacharyya distance for two distributions (Bhattacharyya, 1943):

$$b(p, q) = \sum_i \sqrt{p_i q_i}$$

These results suggest the following **Algorithm 1** to optimize d (and d'):

```
• Set all $\theta_i$ randomly
• Repeat until change in $d$ is very small:
    – Compute $\hat{P}$ according to the above formula
    – For $i$ from 1 to $n$
        ∗ Set $\theta_i$ in such a way as to minimize $D(\hat{P}(X)||P_i(X|\theta_i))$
    – Compute $d$ according to the above formula
```

Each step of the algorithm minimizes d. It is also easy to see that minimizing $D(\hat{P}(X)||P_i(X|\theta_i))$ is the same as setting the parameters θ_i in order to maximize $\prod_{x \in X} P_i(x|\theta_i)^{\hat{P}(x)}$, which can be interpreted as maximizing the probability under P_i of a corpus in which word x appears $\hat{P}(x)$ times. In other words, we are now optimizing $P_i(X)$ given an observed corpus of X, which is a much easier problem. In many types of models for P_i the Expectation-Maximization algorithm is able to solve this problem.

3.2 Combining Pairwise Models

Following the methods outlined in the previous section, we can find an optimal joint probability $P(x_1 \ldots x_n)_{x_i \in L_i}$ if we are given several models $P_j(x_1 \ldots x_n|\theta_j)$. Instead, we have a number of pairwise models. Depending on which independence assumptions we make, we can define a joint distribution over all the languages in various ways. For example, for three languages, A, B, and C, and we can use the following set of models:

$$\begin{aligned} P_1(A, B, C) &= P(A|B)P(B|C)P(C) \\ P_2(A, B, C) &= P(C|A)P(A|B)P(B) \\ P_3(A, B, C) &= P(B|C)P(C|A)P(A) \end{aligned}$$

and

$$\begin{aligned} & d'(\hat{P}, P_1, P_2, P_3) \\ &= D(\hat{P}||P_1) + D(\hat{P}||P_2) + D(\hat{P}||P_3) \\ &= 2H(\hat{P}(A, C), P(A, C)) \\ &+ 2H(\hat{P}(A, B), P(A, B)) \\ &+ 2H(\hat{P}(B, C), P(B, C)) - 3H(\hat{P}) \\ &- H(\hat{P}(A), P(A)) - H(\hat{P}(B), P(B)) \\ &- H(\hat{P}(C), P(C)) \end{aligned}$$

where $H(\cdot)$ is entropy, $H(\cdot, \cdot)$ is cross-entropy, and $\hat{P}(A, B)$ means $\hat{P}$ marginalized to variables A, B. The last three cross-entropy terms involve monolingual models which are not parameterized. The entropy term does not involve any of the pairwise distributions. Therefore, if $\hat{P}$ is fixed, to maximize d'

we need to maximize each of the bilingual cross-entropy terms.

This means we can apply the algorithm from the previous section with a small modification (**Algorithm 2**):

- Set all θ_{ij} (for each language pair i, j) randomly

- Repeat until change in d is very small:
  - Compute P_i for $i = 1 \dots k$ where k is the number of the joint models we have chosen
  - Compute $\hat{P}$ from $\{P_i\}$
  - For i, j such that $i \neq j$
    - Marginalize $\hat{P}$ to (L_i, L_j)
    - Set θ_{ij} in such a way as to minimize $D(\hat{P}(L_i, L_j)||P_i(L_i, L_j|\theta_{ij}))$
  - Compute d according to the above formula

Most of the θ parameters in our models can be set by performing EM, and the rest are discrete with only a few choices and can be maximized over by trying all combinations of them.

4 Building Pairwise Models

We now know how to combine pairwise translation models with some free parameters. Let us now discuss how such models might be built.

Our goal at this stage is to take a parallel bitext in related languages A and B and produce a joint probability model $P(x, y)$, where $x \in A, y \in B$. Equivalently, since the models $P_A(x)$ and $P_B(y)$ are easily estimated by maximum likelihood techniques from the bitext, we can estimate $P_{A\to B}(y|x)$ or $P_{B\to A}(x|y)$. Without loss of generality, we will build $P_{A\to B}(y|x)$.

The model we are building will have a number of free parameters. These parameters will be set by the algorithm discussed above. In this section we will assume that the parameters are fixed.

Our model is a mixture of several components, each discussed in a separate section below:

$$\begin{aligned} P_{A\to B}(y|x) &= \lambda_{fw}(x)P_{fwA\to B}(y|x) \\ &+ \lambda_{bw}(x)P_{bwA\to B}(y|x) \\ &+ \lambda_{char}(x)P_{charA\to B}(y|x) \\ &+ \lambda_{pref}(x)P_{prefA\to B}(y|x) \\ &+ \lambda_{suf}(x)P_{sufA\to B}(y|x) \\ &+ \lambda_{cons}(x)P_{consA\to B}(y|x) \end{aligned} \quad (1)$$

where all λs sum up to one. The λs are free parameters, although to avoid over-training we tie the λs for x's with similar frequencies. These lambdas form a part of the θ_{ij} parameter mentioned previously, where $L_i = A$ and $L_j = B$.

The components represent various constraints that are likely to hold between related languages.

4.1 GIZA (forward)

This component is in fact GIZA++ software, originally created by John Hopkins University's Summer Workshop in 1999, improved by Och (2000). This software can be used to create word alignments for sentence-aligned parallel corpora as well as to induce a probabilistic dictionary for this language pair.

The general approach taken by GIZA is as follows. Let L_A and L_B be the portions of the parallel text in languages A and B respectively, and $L_A = (x_i)_{i=1\dots n}$ and $L_B = (y_i)_{i=1\dots m}$. We can define $P(L_B|L_A)$ as

$$\max_{P_{A\to B}} \max_{P_{\text{aligns}}} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{A\to B}(y_j|x_i)\, P_{\text{aligns}}(x_i|j)$$

The GIZA software does the maximization by building a variety of models, mostly described by Brown et al. (1993). GIZA can be tuned in various ways, most importantly by choosing which models to run and for how many iterations. We treat these parameters as free, to be set along with the rest at a later stage.

As a side effect of GIZA's optimization, we obtain the $P_{A\to B}(y|x)$ that maximizes the above expression. It is quite reasonable to believe that a model of this sort is also a good model for our purposes. This model is what we refer to as $P_{fwA\to B}(y|x)$ in the model overview.

GIZA's approach is not, however, perfect. GIZA builds several models, some quite complex, yet it does not use all the information available to it, notably the lexical similarity between the languages. Furthermore, GIZA tries to map words (especially rare ones) into other words if possible, even if the sentence has no direct translation for the word in question.

These problems are addressed by using other models, described in the following sections.

4.2 GIZA (backward)

In the previous section we discussed using GIZA to try to optimize $P(L_B|L_A)$. It is, however, equally reasonable to try to optimize $P(L_A|L_B)$ instead. If we do so, we can obtain $P_{fwB\to A}(x|y)$ that produces maximal probability for $P(L_A|L_B)$. We, however need a model of $P_{A\to B}(y|x)$. This is easily obtained by using Bayes' rule:

$$P_{bwA\to B}(y|x) = \frac{P_{fwB\to A}(x|y)P_B(y)}{P_A(x)}$$

which requires us to have $P_B(y)$ and $P_A(x)$. These models can be estimated directly from L_B and L_A, by using maximum likelihood estimators:

$$P_A(x) = \frac{\sum_i \delta(x_i, x)}{n}$$

and

$$P_B(y) = \frac{\sum_i \delta(y_i, y)}{m}$$

where $\delta(x, y)$ is the Kronecker's delta function, which is equal to 1 if its arguments are equal, and to 0 otherwise.

4.3 Character-based model

This and the following models all rely on having a model of $P_{A\to B}(y|x)$ to start from. In practice it means that this component is estimated following the previous components and uses the models they provide as a starting point.

The basic idea behind this model is that in related languages words are also related. If we have a model P_c of translating characters in language A into characters in language B, we can define the model for translating entire words.

Let word x in language A consists of characters x_1 through x_n, and word y in language B consist of characters y_1 through y_m.

Let us define (the unnormalized) character model:

$$P_{uchar}(y|x) = P_{charlen}(y|x, m)P_{length}(m|x)$$

i.e., estimating the length of y first, and y itself afterward. We make an independence assumption that the length of y depends only on length of x, and are able to estimate the second term above easily. The first term is harder to estimate.

First, let us consider the case where lengths of x and y are the same ($m = n$). Then,

$$P_{charlen}(y|x, n) = \prod_{i=1}^{n} P_c(y_i|x_i)$$

Let y^j be word y with j's character removed. Let us now consider the case when $m > n$. We define (recursively):

$$P_{charlen}(y|x, m) = \sum_{i=1}^{m} \frac{1}{m} P_{charlen}(y^i|x, m-1)$$

Similarly, if $n > m$:

$$P_{charlen}(y|x) = \sum_{i=1}^{n} \frac{1}{n} P_{charlen}(y|x^i, m)$$

It is easy to see that this is a valid probability model over all sequences of characters. However, y is not a random sequence of characters, but a word in language B, moreover, it is a word that can serve as a potential translation of word x. So, to define a proper distribution over words y given a word x and a set of possible translations of x, $T(x)$

$$\begin{aligned} P_{char}(y|x) &= P_{uchar}\left(y|x, y \in T(x)\right) \\ &= \delta_{y' \in T(x)} \frac{P_{uchar}(y, y \in T(x)|x)}{\sum_{y' \in T(x)} P_{uchar}(y'|x)} \end{aligned}$$

This is the complete definition of P_{char}, except for the fact that we are implicitly relying upon the character-mapping model, P_c, which we need to somehow obtain. To obtain it, we rely upon GIZA again. As we have seen, GIZA can find a good word-mapping model if it has a bitext to work from. If we have a $P_{A\to B}$ word-mapping model of some sort, it is equivalent to having a parallel bitext with words y and x treated as a sequence of characters, instead of indivisible tokens. Each (x, y) word pair would occur $P_{A\to B}(x, y)$ times in this corpus. GIZA would then provide us with the P_c model we need, by optimizing the probability B language part of the model given the language A part.

4.4 Prefix Model

This model and the two models that follow are built on the same principle. Let there be a function $f : A \rightarrow C_A$ and a function $g : B \rightarrow C_B$. These functions group words in A and B into some finite set of classes. If we have some $P_{A\rightarrow B}(y|x)$ to start with, we can define

$$\begin{aligned} & P_{fgA\rightarrow B}(y|x) \\ = \;& P\left(y|g(y)\right) P\left(g(y)|f(x)\right) P\left(f(x)|x\right) \\ = \;& P(y) \frac{\sum_{x':f(x')=f(x)} \sum_{y':g(y')=g(y)} P(x',y')}{\left(\sum_{x':f(x')=f(x)} P(x')\right)\left(\sum_{y':g(y')=g(y)} P(y')\right)} \end{aligned}$$

For the prefix model, we rely upon the following idea: words that have a common prefix often tend to be related. Related words probably should translate as related words in the other language as well. In other words, we are trying to capture word-level semantic information. So we define the following set of f and g functions:

$$f_n(x) = \text{prefix}(x, n)$$

$$g_m(y) = \text{prefix}(y, m)$$

where n and m are free parameters, whose values we will determine later. We therefore define $P_{prefA\rightarrow B}$ as P_{fg} with f and g specified above.

4.5 Suffix Model

Similarly to a prefix model mentioned above, it is also useful to have a suffix model. Words that have the same suffixes are likely to be in the same grammatical case or share some morphological feature which may persist across languages. In either case, if a strong relationship exists between the resulting classes, it provides good evidence to give higher likelihood to the word belonging to these classes. It is worth noting that this feature (unlike the previous one) is unlikely to be helpful in a setting where languages are not related.

The functions f and g are defined based on a set of suffixes S_A and S_B which are learned automatically. $f(x)$ is defined as the longest possible suffix of x that is in the set S_A, and g is defined similarly, for S_B.

The sets S_A and S_B are built as follows. We start with all one-character suffixes. We then consider two-letter suffixes. We add a suffix to the list if it occurs much more often than can be expected based on the frequency of its first letter in the penultimate position, times the frequency of its second letter in the last position. We then proceed in a similar way for three-letter suffixes. The threshold value is a free parameter of this model.

4.6 Constituency Model

If we had information about constituent boundaries in either language, it would have been useful to make a model favoring alignments that do not cross constituent boundaries. We do not have this information at this point. We can assume, however, that any sequence of three words is a constituent of sorts, and build a model based on that assumption.

As before, let $L_A = (x_i)_{i=1...n}$ and $L_B = (y_i)_{i=1...m}$. Let us define as $C_A(i)$ a triple of words (x_{i-1}, x_i, x_{i+1}) and as $C_B(j)$ a triple (y_{j-1}, y_j, y_{j+1}). If we have some model $P_{A\rightarrow B}$, we can define

$$\begin{aligned} P_{C_A\rightarrow C_B}(j|i) &= \tfrac{1}{C} P_{A\rightarrow B}(y_{j-1}|x_{i-1}) P_{A\rightarrow B}(y_j|x_i) \\ &\times \; P_{A\rightarrow B}(y_{j+1}|x_{i+1}) \end{aligned}$$

where C is the sum over j of the above products, and serves to normalize the distribution.

$$\begin{aligned} & P_{consA\rightarrow B}(y|x) \\ = \;& \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} P(y|C_B(j)) P_{C_A\rightarrow C_B}(j|i) P(C_A(i)|x) \\ = \;& \textstyle\sum_{i:x_i=x} \sum_{j=1}^{m} P(y|C_B(j)) P_{C_A\rightarrow C_B}(j|i) \\ = \;& \frac{1}{\sum_{j=1} \delta(y_j, y)} \textstyle\sum_{i:x_i=x} \sum_{j:y_j=y} P_{C_A\rightarrow C_B}(j|i) \end{aligned}$$

5 Evaluation

The output of the system so far is a multi-lingual word translation model. We will evaluate it by producing a tri-lingual dictionary (Russian-Ukrainian-Belorussian), picking a highest probability translation for each word, from the corresponding Bibles. Unfortunately, we do not have a good hand-built tri-lingual dictionary to compare it to, but only one good bilingual one, Russian-Ukrainian[5]. We will therefore take the Russian-Ukrainian portion of our dictionary and compare it to the hand-built one.

Our evaluation metric is the number of entries that match between these dictionaries. If a word has several translations in the hand-built dictionary, match-

[5]The lack of such dictionaries is precisely *why* we do this work

ing any of them counts as correct. It is worth noting that for all the dictionaries we generate, the total number of entries is the same, since all the words that occur in the source portion of the corpus have an entry. In other words, precision and recall are proportional to each other and to our evaluation metric.

Not all of the words that occur in our dictionary occur in the hand-built dictionary and vice versa. An absolute upper limit of performance, therefore, for this evaluation measure is the number of left-hand-side entries that occur in both dictionaries.

In fact, we cannot hope to achieve this number. First, because the dictionary translation of the word in question might never occur in the corpus. Second, even if it does, but never co-occurs in the same sentence as its translation, we will not have any basis to propose it as a translation.[6]. Therefore we have a "achievable upper limit", the number of words that have their "correct" translation co-occur at least once. We will compare our performance to this upper limit.

Since there is no manual tuning involved we do not have a development set, and use the whole bible for training (the dictionary is used as a test set, as described above).

We evaluate the performance of the model with just the GIZA component as the baseline, and add all the other components in turn. There are two possible models to evaluate at each step. The pairwise model is the model given in equation 1 under the parameter setting given by Algorithm 2, with Belorussian used as a third language. The joint model is the full model over these three languages as estimated by Algorithm 2. In either case we pick a highest probability Ukrainian word as a translation of a given Russian word.

The results for Russian-Ukrainian bibles are presented in Table 1. The "oracle" setting is the setting obtained by tuning on the test set (the dictionary). We see that using a third language to tune works just as well, obtaining the true global maximum for the model. Moreover, the joint model (which is more flexible than the model in Equation 1) does even better. This was unexpected for us, because the joint model relies on three pairwise models equally, and Russian-Belorussian and Ukrainian-Belorussian models are bound to be less reliable for Russian-Ukrainian evaluation. It appears, however, that our Belorussian bible is translated directly from Russian rather than original languages, and parallels Russian text more than could be expected.

[6]Strictly speaking, we might be able to infer the word's existence in some cases, by performing morphological analysis and proposing a word we have not seen, but this seems too hard at the moment

Table 1: Evaluation for Russian-Ukrainian (with Belorussian to tune)

Stage	**Pair**	**Joint**
Forward (baseline)	62.3%	71.7%
Forward+chars	77.1%	84.2%
Forward+chars+backward	81.3%	84.1%
Fw+chars+bw+prefix	83.5%	84.5%
Fw+chars+bw+prefix+suffix	84.5%	85%
Fw+chars+bw+pref+suf+const	84.5%	85.2%
"Oracle" setting for λ's	84.6%	

Table 2: Evaluation for Russian-Ukrainian (with Belorussian and Polish)

Tuned by	**Pair**	**Joint**
Belorussian (prev. table)	84.5%	85.2% &
Polish	84.6%	78.6%
Both	84.5%	85.2%
"Oracle" tuning	84.5%	

To insure our results are not affected by this fact we also try Polish separately and in combination with Belorussian (i.e. a model over 4 languages), as shown in Table 2.

These results demonstrate that the joint model is not as good for Polish, but it still finds the optimal parameter setting. This leads us to propose the following extension: let us marginalize joint Russian-Ukrainian-Belorussian model into just Russian-Ukrainian, and add this model as yet another component to Equation 1. Now we cannot use Belorussian as a third language, but we can use Polish, which we know works just as well for tuning. The resulting performance for the model is **85.7%**, our best result to date.

6 Discussion and Future Work

We have built a system for multi-dictionary induction from parallel corpora which significantly improves quality over the standard existing tool (GIZA) by taking advantage of the fact that languages are related and we have a group of more than two of them. Because the system attempts to be completely agnostic about the languages it works on, it might be used successfully on many language groups, requiring almost no linguistic knowledge on the part of the user. Only the prefix and suffix components are somewhat language-specific, but even they are sufficiently general to work, with varying degree of success, on most inflective and agglutinative languages (which form a large majority of languages). For generality, we would also need a model of infixes, for languages such as Hebrew or Arabic. We must admit, however, that we have not tested our approach on other language families yet. It is our short term plan to test our model on several Romance languages, e.g. Spanish, Portuguese, French.

Looking at the first lines of Table 1, one can see that using more than a pair of languages with a model using only a small feature set can dramatically improve performance (compare second and third columns), while able to find the optimal values for all internal parameters.

As discussed in the introduction, the ultimate goal of this project is to produce tools, such as a parser, for languages which lack them. Several approaches are possible, all involving the use of the dictionary we built. While working on this project, we would no longer be treating all languages in the same way. We would use the tools available for that language to further improve the performance of pairwise models involving that language and, indirectly, even the pairs not involving this language. Using these tools, we may be able to improve the word translation model even further, simply as a side effect.

Once we build a high-quality dictionary for a special domain such as the Bible, it might be possible to expand to a more general setting by mining the Web for potential parallel texts.

Our technique is limited in the coverage of the resulting dictionary which can only contain words which occur in our corpus. Whatever the corpus may be, however, it will include the most common words in the target language. These are the words that tend to vary the most between related (and even unrelated) languages. The relatively rare words (e.g. domain-specific and technical terms) can often be translated simply by inferring morphological rules transforming words of one language into another. Thus, one may expand the dictionary coverage using non-parallel texts in both languages, or even in just one language if its morphology is sufficiently regular.

References

The Central Intelligence Agency. 2004. The world factbook.

A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109.

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.

J. Hajic, J. Hric, and V. Kubon. 2000. Machine translation of very close languages. In *Proccedings of the 6th Applied Natural Language Processing Conference*.

P. Koehn and K. Knight. 2001. Knowledge sources for word-level translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

G. Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA*, pages 103–110.

G. Mann and D. Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA*, pages 151–158.

I. D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26:221–249, June.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hongkong, China, October.

F. J. Och and H. Ney. 2001. Statistical multi-source translation. In *Proccedings of MT Summit VIII*, pages 253–258.

Exploiting a Verb Lexicon in Automatic Semantic Role Labelling

Robert S. Swier and **Suzanne Stevenson**
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 3G4
{swier,suzanne}@cs.toronto.edu

Abstract

We develop an unsupervised semantic role labelling system that relies on the direct application of information in a predicate lexicon combined with a simple probability model. We demonstrate the usefulness of predicate lexicons for role labelling, as well as the feasibility of modifying an existing role-labelled corpus for evaluating a different set of semantic roles. We achieve a substantial improvement over an informed baseline.

1 Introduction

Intelligent language technologies capable of full semantic interpretation of domain-general text remain an elusive goal. However, statistical advances have made it possible to address core pieces of the problem. Recent years have seen a wealth of research on one important component of semantic interpretation—automatic role labelling (e.g., Gildea and Jurafsky, 2002; Pradhan et al., 2004; Hacioglu et al., 2004, and additional papers from Carreras and Marquez, 2004). Such work aims to annotate each constituent in a clause with a semantic tag indicating the role that the constituent plays with respect to the target predicate, as in (1):

(1) [Yuka]$_{Agent}$ [whispered]$_{Pred}$ to [Dar]$_{Recipient}$

Semantic role labelling systems address a crucial first step in the automatic extraction of semantic relations from domain-general text, taking us closer to the goal of comprehensive semantic mark-up.

Most work thus far on domain-general role labelling depends on supervised learning over statistical features extracted from a hand-labelled corpus. The reliance on such a resource—one in which the arguments of each predicate are manually identified and assigned a semantic role—limits the portability of such methods to other languages or even to other genres of corpora.

In this study, we explore the possibility of using a verb lexicon, rather than a hand-labelled corpus, as the primary resource in the semantic role labelling task. Perhaps because of the focus on what can be gleaned from labelled data, existing supervised approaches have made little use of the additional knowledge available in the predicate lexicon associated with the labelled corpus. By contrast, we exploit the explicit knowledge of the role assignment possibilities for each verb within an existing lexicon. Moreover, we utilize a very simple probability model within a highly efficient algorithm.

We use VerbNet (Kipper et al., 2000), a computational lexicon which lists the possible semantic role assignments for each of its verbs. Our algorithm extracts automatically parsed arguments from a corpus, and assigns to each a list of the compatible roles according to VerbNet. Arguments which are given only a single role possibility are considered to have been assigned an unambiguous role label. This set of arguments constitutes our *primary-labelled* data, which serves as the noisy training data for a simple probability model which is then used to label the remaining (role ambiguous) arguments.

This method has several advantages, the foremost of which is that it eliminates the dependence on a role labelled corpus, a very expensive resource to produce. Of course, a verb lexicon is also an expensive resource, but one that is highly reusable across a range of NLP tasks. Moreover, the approach points at some potentially useful information that current supervised methods have failed to exploit. Even if one has access to an annotated corpus for training, our work shows that directly calling on additional information from the lexicon itself may prove useful in restricting the possible labels for an argument.

The method has disadvantages as well. The information available in a predicate lexicon is less directly applicable to building a learning model. Inevitably, our results are noisier than in a supervised approach which has access to a labelled sample of what it must produce. Still, the method shows promise: on unseen test data, the system yields an F-measure of .83 on labelling of correctly extracted arguments, compared to an informed baseline of .74, and an F-measure of .65 (compared to .52) on the overall identification and labelling task. The latter is well below the best supervised performance of about .80 on similar tasks, but it must be emphasized that it is achieved with a simple probability model and without the use of hand-labelled data. We view this as a starting point by which to demonstrate the utility of deriving more explicit knowledge from a predicate lexicon, which can be later extended through the use of additional probabilistic features.

We face a methodological challenge arising from the particular choice of VerbNet for the prototyping of our method: the lexicon has no associated semantic role labelled corpus. While this underscores the need for approaches which do not rely on such a resource, it also means that we lack a labelled sample of data against which to evaluate our results. To address this, we use the existing labelled corpus of FrameNet (Baker et al., 1998), and develop a mapping for converting the FrameNet roles to corresponding VerbNet roles. Our mapping method demonstrates the possibility of leveraging existing resources to support the development of role labelling systems based on verb lexicons that do not have an associated hand-labelled corpus.

2 VerbNet Roles and the Role Mapping

Before describing our labelling algorithm, we first briefly introduce the semantic role information available in VerbNet, and describe how we map FrameNet roles to VerbNet roles.

whisper
Frames:
  Agent V
  Agent V Prep(+dest) Recipient
  Agent V Topic
Verbs in same (sub)class:
  [bark, croon, drone, grunt, holler, ...]

Figure 1: A portion of a VerbNet entry.

2.1 The VerbNet Lexicon

VerbNet is a manually developed hierarchical lexicon based on the verb classification of Levin (1993). For each of almost 200 classes containing a total of 3000 verbs, VerbNet specifies the syntactic frames along with the semantic role assigned to each argument position of a frame.[1] Figure 1 shows an example VerbNet entry. The thematic roles used in VerbNet are more general than the situation-specific roles of FrameNet. For example, the roles Speaker, Message, and Addressee of a Communication verb such as *whisper* in FrameNet would be termed Agent, Topic, and Recipient in VerbNet. These coarser-grained roles are often assumed in linguistic theory, and have some advantages in terms of capturing commonalities of argument relations across a wide range of predicates.

2.2 Mapping FrameNet to VerbNet Roles

As noted, VerbNet lacks a corpus of example role assignments against which to evaluate a role labelling based upon it. We create such a resource by adapting the existing FrameNet corpus. We formulate a mapping between FrameNet's larger role set and VerbNet's much smaller one, and create a new corpus with our mapped roles substituted for the original roles in the FrameNet corpus.

We perform the mapping in three steps. First we use an existing mapping between the semantically-specific roles in FrameNet and a much smaller intermediate set of 39 semantic roles which subsume all FrameNet roles.[2] The associations in this mapping are straightforward—e.g., the Place role for Abusing verbs and the Area role for Operate-vehicle verbs are both mapped to Location.

[1]Throughout the paper we use the term "frame" to refer to a syntactic frame—a configuration of syntactic arguments of a verb—possibly labelled with roles, as in Figure 1.

[2]This mapping was provided by Roxana Girju, UIUC.

Second, from this intermediate set we create a simple mapping to the set of 22 VerbNet roles. Some roles are unaffected by the mapping (e.g., Cause alone in the intermediate set maps to Cause in the VerbNet set). Other roles are merged (e.g., Degree and Measure both map to Amount). Moreover, some roles in FrameNet (and the intermediate set) must be mapped to more than one VerbNet role. For example, an Experiencer role in FrameNet is considered Experiencer by some VerbNet classes, but Agent by others. In such cases, our mappings in this step must be specific to the VerbNet class.

In this second step, some roles have no subsuming VerbNet role, because FrameNet provides roles for a wider variety of relations. For example, both FrameNet and the intermediate role set contain a Manner role, which VerbNet does not have. We create a catch-all label, "NoRole," to which we map eight such intermediate roles: Condition, Manner, Means, Medium, Part-Whole, Property, Purpose, and Result. These phrases labelled NoRole are adjuncts—constituents not labelled by VerbNet.

In the third step of our mapping, some of the roles in VerbNet—such as Theme and Topic, Asset and Amount—which appear to be too-fine grained for us to distinguish reliably, are mapped to a more coarse-grained set of VerbNet roles. The final set consists of 16 roles: Agent, Amount, Attribute, Beneficiary, Cause, Destination, Experiencer, Instrument, Location, Material, Predicate, Recipient, Source, Stimulus, Theme and Time; plus the NoRole label.

3 The Frame Matching Process

A main goal of our system is to demonstrate the usefulness of predicate lexicons for the role labelling task. The primary way that we apply the knowledge in our lexicon is via a process we call *frame matching*, adapted from Swier and Stevenson (2004). The automatic frame matcher aligns arguments extracted from an automatically parsed sentence with the frames in VerbNet for the target verb in the sentence. The output of this process is a highly constrained set of candidate roles (possibly of size one) for each potential argument. The resulting singleton sets constitute a (noisy) role assignment for their corresponding arguments, forming our primary-labelled data. This data is then used to train a probability model, described in Section 4, which we employ to label the remaining arguments (those having more than one candidate role).

3.1 Initialization of Candidate Roles

The frame matcher construes extracted arguments from the parsed sentence as being in one of the four main types of syntactic positions (or *slots*) used by VerbNet frames: subject, object, indirect object, and PP-object.[3] Additionally, we specialize the latter by the individual preposition, such as "object of *for*." For the first three slot types, alignment between the extracted arguments and the frames is relatively straightforward. An extracted subject would be aligned with the subject position in a VerbNet frame, for instance, and the subject role from the frame would be listed as a possible label for the extracted subject.

The alignment of PP-objects is similar to that of the other slot types, except that we add an additional constraint that the associated prepositions must match. For PP-object slots, VerbNet frames often provide an explicit list of allowable prepositions. Alternatively, the frame may specify a required semantic feature such as `+path` or `+loc`. In order for an extracted PP-object to align with one of these frame slots, its associated preposition must be included in the list provided by the frame, or have the specified feature. To determine the latter, we manually create lists of prepositions that we judge to have each of the possible semantic features.

In general, this matching procedure assumes that frames describing a syntactic argument structure similar to that of the parsed sentence are more likely to correctly describe the semantic roles of the extracted arguments. Thus, the frame matcher only chooses roles from frames that are the best syntactic matches with the extracted argument set. This is achieved by adopting the scoring method of Swier and Stevenson (2004), in which we compute the portion *%Frame* of frame slots that can be mapped to an extracted argument, and the portion *%Sent* of extracted arguments from the sentence that can be mapped to the frame. The score for each frame is given by *%Frame*+*%Sent*, and only frames having the highest score contribute candidate roles to the

[3]Since VerbNet has very few verbs with sentential complements, we do not consider them for now.

Possible Frames for Verb V	Extracted Slots SUBJ	OBJ	%Frame	%Sent	Score
Agent V	Agent		100	50	150
Agent V Theme	**Agent**	**Theme**	**100**	**100**	**200**
Instrument V Theme	**Instrument**	**Theme**	**100**	**100**	**200**
Agent V Recipient Theme	Agent	Theme	67	100	167

Table 1: An example of frame matching.

extracted arguments. An example scoring is shown in Table 1. Note that two of the frames are tied for the highest score of 200, resulting in two possible roles for the subject (Agent and Instrument), and Theme as the only possible role for the object.

As mentioned, this frame matching step is very restrictive, and it greatly reduces role ambiguity. Many potential arguments receive only a single candidate role, providing the primary-labelled data we use to train our probability model. Some slots receive *no* candidate roles, which is an error for argument slots but which is correct for adjuncts. The reduction of candidate roles in general is very helpful in lightening the subsequent load on the probability model to be applied next, but note that it may also cause the correct role to be omitted. We experiment with choosing roles from the frames that are the best syntactic matches, and from all possible frames.

3.2 Adjustments to the Role Mapping

We further extend the frame matcher, which has extensive knowledge of VerbNet, for the separate task of helping to eliminate some of the inconsistencies that are introduced by our role mapping procedure. This is a process that applies concurrently with the initialization of candidate roles described above, but only affects the gold standard labelling of evaluation data.[4]

For instance, FrameNet assigns the role Side2 to the object of the preposition *with* occurring with the verb *brawl*. Side2 is mapped to Theme by our role mapping; however, in VerbNet, *brawl* does not accept Theme as the object of *with*. Our mapping thus creates a target (i.e., gold standard) label in the evaluation data that is inconsistent with VerbNet. Since there is no possibility of the role labeller assigning a label that matches such a target, this unfairly raises the task difficulty. However, since *brawl* does accept Theme in another slot, it is not an option to entirely eliminate this role in the mapping for the verb. Instead, we use our frame matcher to verify that each target role generated by our mapping from FrameNet is allowed by VerbNet in the relevant slot. If the target role is not allowed, then it is converted to NoRole in the evaluation set. Constituents labelled as NoRole are not considered target arguments, and it is correct for the system to not assign labels in these cases.

The NoRole conversions help to ensure that our gold standard evaluation data is consistent with our lexicon, but the method does have limitations. For instance, some of the arguments which the system fails to extract might have had their target role changed to NoRole if they were properly extracted. Additionally, in some cases a target role is converted to NoRole when there is an actual role that VerbNet would have assigned instead.

4 The Probability Model

Once argument slots are initialized with sets of possible roles, the algorithm uses a probability model to label slots having two or more possibilities. Since our primary goal is to demonstrate how much can be accomplished through the frame matcher, we compare a number of very simple probability models:

- $\mathbf{P}(\mathbf{r}|\mathbf{v},\mathbf{s})$: the probability of a role given the target verb and the slot; the latter includes subject, object, indirect object, and prepositional object, where each PP slot is specialized by the identity of the preposition;
- $\mathbf{P}(\mathbf{r}|\mathbf{s})$: the probability of a role given the slot;
- $\mathbf{P}(\mathbf{r}|\mathbf{sc})$: the probability of a role given the slot class, in which all prepositional slots are treated together.

[4]Of course, the fact that the frame matcher "sees" the evaluation set as part of its dual duties is not allowed to influence its assignment of candidate roles.

Each probability model predicts a role given certain conditioning information, with maximum likelihood estimates determined by the primary-labelled data directly resulting from the frame matching step.[5]

We also compare one non-probabilistic model to resolve the same set of ambiguous cases:

- **Default assignment**: candidate roles for ambiguous slots are ignored; the four slot classes of subject, object, indirect object and PP-object are assigned the roles Agent, Theme, Recipient, and Location, respectively.

These are the most likely roles assigned by the frame matcher over our development data.

For comparison, we also apply the iterative algorithm developed by Swier and Stevenson (2004), using the same bootstrapping parameters. The method uses backoff over three levels of specificity of probabilities.

5 Materials and Methods

5.1 The Target Verbs

For ease of comparison, we use the same verbs as in Swier and Stevenson (2004), except that we measure performance over a much larger superset of verbs. In that work, a core set of 54 target verbs are selected to represent a variety of classes with interesting role ambiguities, and the system is evaluated against only those verbs. An additional 1105 verbs—all verbs sharing at least one class with the target verbs—are also labelled, in order to provide more data for the probability estimations. Here, we consider our system's performance over the 1159 target verbs that consist of the union of these two sets of verbs.

5.2 The Corpus and Preprocessing

The majority of sentences in FrameNet II are taken from the British National Corpus (BNC Reference Guide, 2000). Our development and test data consists of a percentage of these sentences. For some experiments, these sentences are then merged with a random selection of additional sentences from the BNC in order to provide more training data for the probability estimations. We evaluate performance only on FrameNet sentences that include our target verbs.

All of our corpus data was parsed using the Collins parser (Collins, 1999). Next, we use TGrep2 (Rohde, 2004) to automatically extract from the parse trees the constituents forming potential arguments of the target verbs. For each verb, we label as the subject the lowest NP node, if it exists, that is immediately to the left of a VP node which dominates the verb. Other arguments are identified by finding sister NP or PP nodes to the right of the verb. Heads of noun phrases are identified using the method of Collins (1999), which primarily chooses the rightmost noun in the phrase that is not inside a prepositional phrase or subordinate clause. Error may be introduced at each step of this preprocessing—the sentence may be misparsed, some arguments (such as distant subjects) may not be extracted, or the wrong word may be found as the phrase head.

5.3 Validation and Test Data

A random selection of 30% of the preprocessed FrameNet data is set aside for testing, and another random 30% is used for development and validation. For experiments involving additional BNC data, each 30% of the FrameNet sentences is embedded in a random selection of 20% of the BNC. We selected these percentages to yield a sufficient amount of data for experimentation, while reserving some unseen data for future work. The FrameNet portion of the validation set includes 515 types of our target verbs (across 161 VerbNet classes) in 4300 sentences, and contains a total of 6636 target constituents—i.e., constituents that receive a valid VerbNet role as their gold standard label, not NoRole. The test set includes 517 of the target verbs (from 163 classes) in 4308 sentences, yielding 6705 target constituents.[6]

To create an evaluation set, we map the manually annotated FrameNet roles in the corpus to VerbNet roles (or NoRole), as described in Sections 2.2 and 3.2. We use this role information to calculate performance: the system should assign roles matching the target VerbNet roles, and make no assignment when the target is NoRole.

[5]Note that we assume the probability of a role for a slot is independent of other slots—that is, we do not ensure a consistent role assignment to all arguments across an instance of a verb.

[6]The verbs appearing in the validation and test sets occur respectively across 161 and 165 FrameNet classes (what in FrameNet are called "frames").

5.4 Methods of Argument Identification

One of the decisions we face is how to evaluate the identification of extracted arguments generated by the system against the manually annotated target arguments provided by FrameNet. We try two methods, the most strict of which is to require full-phrase agreement: an extracted argument and a target argument must cover exactly the same words in the sentence in order for the argument to be considered correctly extracted. This means, for instance, that a prepositional phrase incorrectly attached to an extracted object would render the object incompatible with the target argument, and any system label on it would be counted as incorrect. This evaluation method is commonly used in other work (e.g., Carreras and Marquez, 2004).

The other method we use is to require that only the head of an extracted argument and a target argument match. This latter method helps to provide a fuller picture of the range of arguments found by the system, since there are fewer near-misses caused by attachment errors. Since heads of phrases are often the most semantically relevant part of an argument, labels on heads provide much of the same information as labels on whole phrases. For these reasons, we use head matching for most of our experiments below. For comparison, however, we provide results based on full-phrase matching as well.

6 Experimental Results

6.1 Experimental Setup

We evaluate our system's performance on several aspects of the overall role labelling task; all results are given in terms of F-measure, $2PR/(P+R)$.[7] The first task is argument identification, in which constituents considered by our system to be arguments (i.e., those that are extracted and labelled) are evaluated against actual target arguments. The second task is labelling extracted arguments, which evaluates the labelling of only those arguments that were correctly extracted. Last is the overall role labelling task, which evaluates the system on the combined tasks of identification and labelling of all target arguments.

We compare our results to an informed baseline that has access to the same set of extracted arguments as does the frame matcher. The baseline labels all extracted arguments using the default role assignments described in Section 4.

In addition to experiments in which we employ various methods of resolving ambiguous assignments, we also evaluate the system with varying types and amounts of training data, and with two alternate methods for choosing frames from which to draw candidate roles.

6.2 Evaluation of Probability Models

We first evaluate our system with the three very simple probability models, as well as the non-probabilistic default assignment, to determine roles for the extracted arguments that the frame matcher considers to be ambiguous. We also report results after only the frame matcher has been applied, to indicate how much work is being done by it alone. Because we have constructed the frame matcher to be highly restrictive in assigning candidate roles to extracted arguments, a large number (about 62%) become primary-labelled data and so do not require resolution of ambiguous roles. Only about 16% of our extracted arguments have role ambiguities, and about 22% (many of which are adjuncts) do not receive any candidates and remain unlabelled.

Task:	Id.	Lab.	Id. + Lab.
Baseline	.80	.74	.52
FM + $P(r\|sc)$	.83	.83	.65
FM + $P(r\|s)$	.83	.84	.65
FM + $P(r\|v,s)$	.83	.78	.61
FM + Dflt. Assgnmt.	.83	.82	.64
FM only	.83	.76	.60

As shown in the table, all models perform equally well on identification, which is determined by the frame matcher (FM); i.e., any extracted argument receiving one or more candidate roles is "identified" as an argument. Performance is somewhat above the baseline, which must label *all* extracted arguments. For the task of labelling correctly extracted arguments and for the combined task, the simplest probability models, $P(r|sc)$ and $P(r|s)$, perform about the same. On the combined task, they achieve .13 above the informed baseline, indicating the effectiveness of such simple models when combined with the frame matcher. The more specific model, $P(r|v,s)$, performs less well, and may be over-fitting on this relatively small amount of training data.

[7]In each case, P and R are close in value.

Two observations indicate the power of the frame matcher. First, even using the non-probabilistic default assignments to resolve ambiguous roles substantially outperforms the baseline (and indeed performs quite close to the best results, since the default role assignment is often the same as that chosen by the probability models). Importantly, the baseline uses the *same* default assignments, but without the benefit of the frame matcher to further narrow down the possible arguments. Second, the frame matcher alone achieves .60 F-measure on the combined task, not far below the performance of the best models. These results show that once arguments have been extracted, much of the labelling work is performed by the frame matcher's careful application of lexical information.

Henceforth we consider the use of the frame matcher plus $P(r|sc)$ as our basic system, since this is our simplest model, and no other outperforms it.

6.3 Evaluation of Training Methods

In our above experiments, the probabilistic models are trained only on primary-labelled data from the frame matcher run on the FrameNet data. We would like to determine whether using either more data or less noisy data may improve results. To provide more data, we ran the frame matcher on the additional 20% of the BNC. This provides almost 600K more sentences containing our target verbs, yielding a much higher amount of primary-labelled data. To provide less-noisy data, we trained the probability models on manually annotated target labels from system-identified arguments in 1000 sentences. While fewer sentences are used, all arguments in the training data are guaranteed to have a correct role assignment, in contrast to the primary-labelled data output by the frame matcher. (We chose 1000 sentences as an upper bound on an amount of data that could be relatively easily annotated by human judges.)

Training Data:	Prim.-lab. FN	Prim.-lab. BNC	1K sents annot'd
Baseline	.52		
FM + $P(r\|sc)$	.65	.65	.65
FM + $P(r\|v,s)$	.61	.62	.63

For our basic model, $P(r|sc)$, these variations in training data do not affect performance. Only the most specific model, $P(r|v,s)$, shows improvement when trained on more data or on manually annotated data, although it still does not perform as well as the simplest model. Because the models only choose from among candidate roles selected by the frame matcher, differences in the learned probability estimations must be quite large to have an effect. At least for the simplest model, these estimations do not vary with a larger corpus or one lacking in noise. However, the increase in performance seen here for the more specific model, albeit small, may indicate that richer probability models may require more or cleaner training data.

6.4 Evaluation of Frame Choice

	"Best" frames	All Frames
Baseline	.52	
FM + $P(r\|sc)$	.65	.63

The frame matcher has been shown to shoulder much of the responsibility in our system, and it is worth considering variations in its operation. For example, by having the frame matcher only choose roles from the frames that are the best syntactic matches to the sentence, role ambiguity is minimized at the cost of possibly excluding the correct role. To determine whether we may do better by relying more on the probability model and less on the frame matcher, we instead include role candidates from all frames in a verb's lexical entry. The effect of this choice is more role ambiguity, decreasing the number of primary-labelled slots by roughly 30%. We see that performance using $P(r|sc)$ is slightly worse with the greater ambiguity admitted by using all frames, indicating the benefit of precise selection of candidate roles.

6.5 Differing Argument Evaluation Methods

	Heads	Full Phrase
Baseline	.52	.49
FM + $P(r\|sc)$	.65	.61

As mentioned, for most of our evaluations we match the arguments extracted by the system to the target arguments via a match on phrase heads, since head labels provide much useful semantic information. When we instead require that the extracted arguments match the targets exactly, the number of correctly extracted arguments falls from about 80% of the roughly 6700 targets to about 74%, due to increased parsing difficulty. As expected, this results in both the system and the baseline having performance decreases on the overall task.

7 Related Work

Most role labelling systems have required hand-labelled training data. Two exceptions are the subcategorization frame based work of Atserias et al. (2001) and the bootstrapping labeller of Swier and Stevenson (2004), but both are evaluated on only a small number of verbs and arguments. In related unsupervised tasks, Riloff and colleagues have learned "case frames" for verbs (e.g., Riloff and Schmelzenbach, 1998), while Gildea (2002) has learned role-slot mappings (but does not apply the knowledge for the labelling task).

Other role labelling systems have also relied on the extraction of much more complex features or probability models than we adopt here. As a point of comparison, we apply the iterative backoff model from Swier and Stevenson (2004), trained on 20% of the BNC, with our frame matcher and test data. The backoff model achieves an F-measure of .63, slightly below the performance of .65 for our simplest probability model, which uses less training data and takes far less time to run (minutes rather than hours).

In general, it is not possible to make direct comparisons between our work and most other role labellers because of differences in corpora and role sets, and, perhaps more significantly, differences in the selection of target arguments. However, the best supervised systems, using automatic parses to identify full argument phrases in PropBank, achieve about .82 on the task of identifying and labelling arguments (Pradhan et al., 2004). Though this is higher than our performance of .61 on full phrase arguments, our system does not require manually annotated data.

8 Conclusion

In this work, we employ an expensive but highly reusable resource—a verb lexicon—to perform role labelling with a simple probability model and a small amount of unsupervised training data. We outperform similar work that uses much more data and a more complex model, showing the benefit of exploiting lexical information directly. To achieve performance comparable to that of supervised methods may require human filtering or augmentation of the initial labelling. However, given the expense of producing a large semantically annotated corpus, even such "human in the loop" approaches may lead to a decrease in overall resource demands. We use such a corpus for evaluation purposes only, modifying it with a role mapping to correspond to our lexicon. We thus demonstrate that such existing resources can be bootstrapped for lexicons lacking an associated annotated corpus.

Acknowledgments

We gratefully acknowledge the support of NSERC of Canada. We also thank Afsaneh Fazly, who assisted with much of our corpus pre-processing.

References

J. Atserias, L. Padró, and G. Rigau. 2001. Integrating multiple knowledge sources for robust semantic parsing. In *Proc. of the International Conf. on Recent Advances in NLP*.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The Berkeley FrameNet Project. In *Proc. of COLING-ACL*, p. 86–90.

BNC Reference Guide. 2000. *Reference Guide for the British National Corpus (World Edition)*, second edition.

X. Carreras and L. Marquez, editors. 2004. *CoNLL-04 Shared Task*.

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

D. Gildea. 2002. Probabilistic models of verb-argument structure. In *Proc. of the 19th International CoNLL*, p. 308–314.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 23(3):245–288.

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proc. of the 8th CoNLL*, p. 110–113.

K. Kipper, H. T. Dang, and M. Palmer. 2000. Class based construction of a verb lexicon. In *Proc. of the 17th AAAI Conf.*

B. Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proc. of HLT/NAACL*.

E. Riloff and M. Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proc. of the 6th WVLC*.

D. L. T. Rohde. 2004. TGrep2 user manual ver. 1.11. http://tedlab.mit.edu/˜dr/Tgrep2.

R. Swier and S. Stevenson. 2004. Unsupervised semantic role labelling. In *Proc. of the 2004 Conf. on EMNLP*, p. 95–102.

A Semantic Scattering Model for the Automatic Interpretation of Genitives

Dan Moldovan
Language Computer Corporation
Richardson, TX 75080
moldovan@languagecomputer.com

Adriana Badulescu
Language Computer Corporation
Richardson, TX 75080
adriana@languagecomputer.com

Abstract

This paper addresses the automatic classification of the *semantic relations* expressed by the English genitives. A learning model is introduced based on the statistical analysis of the distribution of genitives' semantic relations on a large corpus. The semantic and contextual features of the genitive's noun phrase constituents play a key role in the identification of the semantic relation. The algorithm was tested on a corpus of approximately 2,000 sentences and achieved an accuracy of 79% , far better than 44% accuracy obtained with C5.0, or 43% obtained with a Naive Bayes algorithm, or 27% accuracy with a Support Vector Machines learner on the same corpus.

1 Introduction

1.1 Problem Description

The identification of semantic relations in open text is at the core of Natural Language Processing and many of its applications. Detecting semantic relations is useful for syntactic and semantic analysis of text and thus plays an important role in automatic text understanding and generation. Furthermore, semantic relations represent the core elements in the organization of lexical semantic knowledge bases used for inferences. Recently, there has been a renewed interest in text semantics fueled in part by the complexity of some major research initiatives in Question Answering, Text Summarization, Text Understanding and others, launched in the United States and abroad.

Two of the most frequently used linguistic constructions that encode a large set of semantic relations are the s-genitives, e.g. "*man's brother*", and the of-genitives, e.g. "*dress of silk*". The interpretation of these phrase-level constructions is paramount for various applications that make use of lexical semantics.

Example: "*The child's mother* had moved the child from a car safety seat to an area near the open *passenger-side door of the car*." (The Desert Sun, Monday, October 18th, 2004).

There are two semantic relations expressed by genitives: (1) "*child's mother*" is an s-genitive encoding a KINSHIP relation, and (2) "*passenger-side door of the car*" is an of-genitive encoding a PART-WHOLE relation.

This paper provides a detailed corpus analysis of genitive constructions and a model for their automatic interpretation in English texts.

1.2 Semantics of Genitives

In English there are two kinds of genitives. In general, in one, the modifier is morphologically linked to the possessive clitic *'s* and precedes the head noun (*s-genitive*, i.e. NP_{modif}*'s* NP_{head}), and in the second one the modifier is syntactically marked by the preposition *of* and follows the head noun (*of-genitive*, i.e. NP_{head} *of* NP_{modif}).

Although the genitive constructions have been studied for a long time in cognitive linguistics, their semantic investigation proved to be very difficult, as the meanings of the two constructions are difficult to pin down. There are many factors that contribute to the genitives' semantic behavior, such as the type of the genitive, the semantics of the constituent nouns, the surrounding context, and others.

A characteristic of genitives is that they are very productive, as the construction can be given various semantic interpretations. However, in some situations, the number of interpretations can be reduced by employing world knowledge. Consider the examples, *"Mary's book"* and *"Shakespeare's book"*. *"Mary's book"* can mean the book Mary owns, the book Mary wrote, the book Mary is reading, or the book Mary is very fond of. Each of these interpretations is possible in the right context. In *"Shakespeare's book"*, however, the preferred interpretation, provided by a world knowledge dictionary, is the book written by Shakespeare.

1.3 Previous Work

There has been much interest recently on the discovery of semantic relations from open-text using symbolic and statistical techniques. This includes the seminal paper of (Gildea and Jurafsky, 2002), Senseval 3 and coNLL competitions on automatic labeling of semantic roles detection of noun compound semantics (Lapata, 2000), (Rosario and Hearst, 2001) and many others. However, not much work has been done to automatically interpret the genitive constructions.

In 1999, Berland and Charniak (Berland and Charniak, 1999) applied statistical methods on a very large corpus to find PART-WHOLE relations. Following Hearst's method for the automatic acquisition of hypernymy relations (Hearst, 1998), they used the genitive construction to detect PART-WHOLE relations based on a list of six seeds representing *whole* objects, (i.e. *book, building, car, hospital, plant, and school*). Their system's output was an ordered list of possible *parts* according to some statistical metrics (Dunning's log-likelihood metric and Johnson's significant-difference metric). They presented the results for two specific patterns (*"NN's NN"* and *"NN of DT NN"*). The accuracy obtained for the first 50 parts was 55% and for the first 20 parts was 70%.

In 2003, Girju, Badulescu, and Moldovan (Girju, Badulescu, and Moldovan, 2003) detected the PART-WHOLE relations for some of the most frequent patterns (including the genitives) using the Iterative Semantic Specialization, a learning model that searches for constraints in the WordNet noun hierarchies. They obtained an f-measure of 93.62% for s-genitives and 91.12% for of-genitives for the PART-WHOLE relation.

Given the importance of the semantic relations encoded by the genitive, the disambiguation of these relations has long been studied in cognitive linguistics (Nikiforidou, 1991), (Barker, 1995), (Taylor, 1996), (Vikner and Jensen, 1999), (Stefanowitsch, 2001), and others.

2 Genitives' Corpus Analysis

2.1 The Data

In order to provide a general model of the genitives, we analyzed the syntactic and semantic behavior of both constructions on a large corpus of examples selected randomly from an open domain text collection, LA Times articles from TREC-9. This analysis is justified by our desire to answer the following questions: "*What are the semantic relations encoded by the genitives?*" and "*What is their distribution on a large corpus?*"

A set of 20,000 sentences were randomly selected from the LA Times collection. In these 20,000 sentences, there were 3,255 genitive instances (2,249 of-constructions and 1,006 s-constructions). From these, 80% were used for training and 20% for testing.

Each genitive instance was tagged with the corresponding semantic relations by two annotators, based on a list of 35 most frequently used semantic relations proposed by (Moldovan et al., 2004) and shown in Table 1. The genitives' noun components were manually disambiguated with the corresponding WordNet 2.0 senses or the named entities if they are not in WordNet (e.g. names of persons, names of locations, etc).

2.2 Inter-annotator Agreement

The annotators, two graduate students in Computational Semantics, were given the genitives and the sentences in which they occurred. Whenever the annotators found an example encoding a semantic relation other than those provided, they had to tag it as "OTHER". Besides the type of relation, the annotators were asked to provide the correct WordNet 2.0 senses of the two nouns and information about the *order* of the modifier and the head nouns in the genitive construction. For example, although in of-constructions the head is followed by the modifier most of the time, this is not always true. For instance, in "*owner of car*[POSSESSION]" the head *owner* is followed by the modifier *car*, while in "*John's car*[POSSESSION/R]" the order is reversed. Approximately one third of the training examples had the nouns in reverse order.

Most of the time, one genitive instance was tagged with one semantic relation, but there were also situations in which an example could belong to more than one relation in the same context. For example, the genitive "*city of USA*" was tagged as a PART-WHOLE relation and as a LOCATION relation. There were 21 such cases in the training corpus.

The judges' agreement was measured using the Kappa statistics (Siegel and Castelan, 1988), one of the most frequently used measure of inter-annotator agreement for classification tasks: $K = \frac{Pr(A)-Pr(E)}{1-Pr(E)}$, where $Pr(A)$ is the proportion of times the raters agree and $Pr(E)$ is the probability of agreement by chance.

The K coefficient is 1 if there is a total agreement among the annotators, and 0 if there is no agreement other than that expected to occur by chance.

On average, the K coefficient is close to 0.82 for both of and s-genitives, showing a good level of agreement for the training and testing data on the set of 35 relations, taking into consideration the task difficulty. This can be explained by the instructions the annotators received prior to annotation and by their expertise in lexical semantics.

2.3 Distribution of Semantic Relations

Table 1 shows the distribution of the semantic relations in the annotated corpus.

In the case of of-genitives, there were 19 relations found from the total of 35 relations considered. The most frequently occurring relations were POSSESSION, KINSHIP, PROPERTY, PART-WHOLE, LOCATION, SOURCE, THEME, and MEASURE.

There were other relations (107 for of-genitives) that do not belong to the predefined list of 35 relations, such as "*state of emergency*". These examples were clustered in different undefined subsets based on their semantics. The largest subsets did not cover more than 3% of the OTHER set of examples. This observation shows that the set of 35 semantic relations from Table 1 is representative for genitives.

No.	Freq.		Semantic Relations	Examples
	Of	S		
1	36	220	POSSESSION	*"Mary's book"*
2	25	61	KINSHIP	*"Mary's brother"*
3	109	75	PROPERTY	*"John's coldness"*
4	11	123	AGENT	*"investigation of the crew"*
5	5	109	TIME-EVENT	*"last year's exhibition"*
6	30	7	DEPICTION-DEPICTED	*"a picture of my nice"*
7	328	114	PART-WHOLE	*"the girl's mouth"*
8	0	0	HYPERNYMY (IS-A)	*"city of Dallas"*
9	0	0	ENTAILMENT	N/A
10	10	3	CAUSE	*"death of cancer"*
11	11	62	MAKE/PRODUCE	*"maker of computer"*
12	0	0	INSTRUMENT	N/A
13	32	46	LOCATION/SPACE	*"university of Texas"*
14	0	0	PURPOSE	N/A
15	56	33	SOURCE/FROM	*"president of Bolivia"*
16	70	5	TOPIC	*"museum of art"*
17	0	0	MANNER	N/A
18	0	0	MEANS	*"service of bus"*
19	10	4	ACCOMPANIMENT	*"solution of the problem"*
20	1	2	EXPERIENCER	*"victim of lung disease"*
21	49	41	RECIPIENT	*"Josephine's reward"*
22	0	0	FREQUENCY	N/A
23	0	0	INFLUENCE	N/A
24	5	2	ASSOCIATED WITH	*"contractors of shipyard"*
25	115	1	MEASURE	*"hundred of dollars"*
26	0	0	SYNONYMY	N/A
27	0	0	ANTONYMY	N/A
28	0	0	PROB. OF EXISTENCE	N/A
29	0	0	POSSIBILITY	N/A
30	0	0	CERTAINTY	N/A
31	120	50	THEME	*"acquisition of the holding"*
32	8	2	RESULT	*"result of the review"*
33	0	0	STIMULUS	N/A
34	0	0	EXTENT	N/A
35	0	0	PREDICATE	N/A
36	107	49	OTHER	*"state of emergency"*

Table 1: The distribution of the semantic relations in the annotated corpus of 20,000 sentences.

Table 1 also shows the semantic preferences of each genitive form. For example, POSSESSION, KINSHIP, and some kinds of PART-WHOLE relations are most of the time encoded by the s-genitive, while some specific PART-WHOLE relations, such as "*dress of silk*" and "*array of flowers*", cannot be encoded but only by the of-genitive. This simple analysis leads to the important conclusion that the two constructions must be treated separately as their semantic content is different. This observation is also consistent with other recent work in linguistics on the grammatical variation of the English genitives (Stefanowitsch, 2001).

3 The Model

3.1 Problem Formulation

Given a genitive, the goal is to develop a procedure for the automatic labeling of the semantic relation it conveys. The semantic relation derives from the semantics of the noun phrases participating in genitives as well as the surrounding context.

Semantic classification of syntactic patterns in general can be formulated as a learning problem. This is a multi-class classification problem since the output can be one of the semantic relations in the set. We cast this as a supervised learning problem where input/ output pairs are available as training data.

An important first step is to map the characteristics of each genitive construction into a feature vector. Let's define with $\mathbf{x}_i$ the feature vector of an instance i and let X be the space of all instances; i.e. $\mathbf{x}_i \in X$. The multi-class classification is performed by a function that maps the feature space X into a semantic space S

$F : X \rightarrow S$, where S is the set of semantic relations from Table 1, i.e. $r_k \in S$.

Let T be the training set of examples or instances $T = (\mathbf{x}_1 r_1, \mathbf{x}_2 r_2, ..., \mathbf{x}_n r_n) \subseteq (X \times S)^n$ where n is the number of examples $\mathbf{x}$ each accompanied by its semantic relation label r. The problem is to decide which semantic relation r to assign to a new, unseen example $\mathbf{x}_{n+1}$. In order to classify a given set of examples (members of X), one needs some kind of measure of the similarity (or the difference) between any two given members of X.

3.2 Feature Space

An essential aspect of our approach below is the word sense disambiguation (WSD) of the noun. Using a state-of-the-art open-text WSD system with 70% accuracy for nouns (Novischi et al., 2004), each word is mapped into its corresponding WordNet 2.0 sense. The disambiguation process takes into account surrounding words, and it is through this process that *context* gets to play a role in labeling the genitives' semantics.

So far, we have identified and experimented with the following NP **features**:

1. **Semantic class of head noun** specifies the WordNet sense (synset) of the head noun and implicitly points to all its hypernyms. It is extracted automatically via a word sense disambiguation module. The genitive semantics is influenced heavily by the meaning of the noun constituents. For example: "*child's mother*" is a KINSHIP relation where as "*child's toy*" is a POSSESSION relation.

2. **Semantic class of modifier noun** specifies the WordNet synset of the modifier noun. The following examples show that the semantic of a genitive is also influenced by the semantic of the modifier noun; "*Mary's apartment*" is a POSSESSION relation, and "*apartment of New York*" is a LOCATION relation.

The positive and negative genitive examples of the training corpus are pairs of concepts of the format:

<modifier_semclass#WNsense;
head_semclass#WNsense; target>,

where `target` is a set of at least one of the 36 semantic relations. The `modifier_semclass` and `head_semclass` concepts are WordNet semantic classes tagged with their corresponding WordNet senses.

3.3 Semantic Scattering Learning Model

For every pair of <modifier - head> noun genitives, let us define with f_i^m and f_j^h the WordNet 2.0 senses of the modifier and head respectively. For convenience we replace the tuple $< f_i^m, f_j^h >$ with f_{ij}. The Semantic Scattering Model is based on the following observations:

Observation 1. f_i^m and f_j^h can be regarded as nodes on some paths that link the senses of the most specific noun concepts with the top of the noun hierarchies.

Observation 2. The closer the pair of noun senses f_{ij} is to the bottom of noun hierarchies the fewer the semantic relations associated with it; the more general f_{ij} is the more semantic relations.

The probability of a semantic relation *r* given feature pair f_{ij}

$$P(r|f_{ij}) = \frac{n(r, f_{ij})}{n(f_{ij})}, \quad (1)$$

is defined as the ratio between the number of occurrences of a relation *r* in the presence of feature pair f_{ij} over the number of occurrences of feature pair f_{ij} in the corpus. The most probable relation $\hat{r}$ is

$$\hat{r} = argmax_{r \in R} P(r|f_{ij}) \quad (2)$$

From the training corpus, one can measure the quantities $n(r, f_{ij})$ and $n(f_{ij})$. Depending on the level of abstraction of f_{ij} two cases are possible:

Case 1. The feature pair f_{ij} is specific enough such that there is only one semantic relation r for which

$P(r|f_{ij}) = 1$ and 0 for all the other semantic relations.

Case 2. The feature pair f_{ij} is general enough such that there are at least two semantic relations for which $P(r|f_{ij}) \neq 0$. In this case equation (2) is used to find the most appropriate $\hat{r}$.

Definition. A boundary G^* in the WordNet noun hierarchies is a set of synset pairs such that :

a) for any feature pair on the boundary, denoted $f_{ij}^{G^*} \in G^*$, $f_{ij}^{G^*}$ maps uniquely into only one relation r, and

b) for any $f_{ij}^u \succ f_{ij}^{G^*}$, f_{ij}^u maps into more than one relation r, and

c) for any $f_{ij}^l \prec f_{ij}^{G^*}$, f_{ij}^l maps uniquely into a semantic relation r. Here relations $\succ$ and $\prec$ mean "semantically more general" and "semantically more specific" respectively. This is illustrated in Figure 1.

Observation 3. We have noticed that there are more concept pairs under the boundary G^* than above, i.e. $| \{f_{ij}^l\} | \gg | \{f_{ij}^u\} |$.

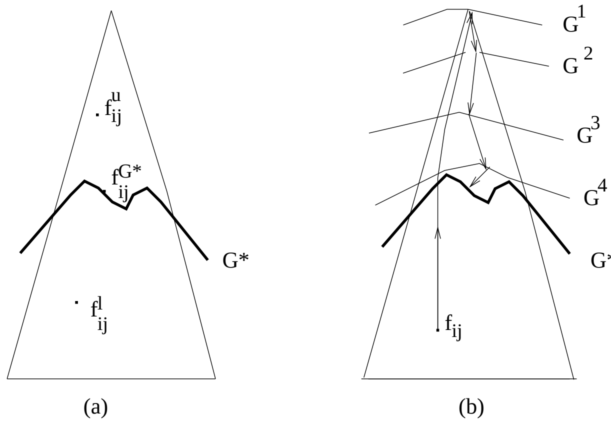


Figure 1: (a) Conceptual view of the noun hierarchies separated by the boundary G^*; (b) Boundary G^* is found through an iterative process called "semantic scattering".

3.4 Boundary Detection Algorithm

An approximation to boundary G^* is found using the training set through an iterative process called *semantic scattering*. We start with the most general boundary corresponding to the nine noun WordNet hierarchies and then specialize it based on the training data until a good approximation is reached.

Step 1. Create an initial boundary

The initial boundary denoted G^1 is formed from combinations of the nine WordNet hierarchies: *abstraction#6, act#2, entity#1, event#1, group#1, possession#2, phenomenon#1, psychological_feature#1, state#4.* To each training example a corresponding feature $f_{ij} =< f_i^m, f_j^h >$ is first determined, after which is replaced with the most general corresponding feature consisting of top WordNet hierarchy concepts denoted with f_{ij}^1. For instance, to the example "*apartment of the woman*" it corresponds the general feature *entity#1-entity#1* and POSSESSION relation, to "*husband of the woman*" it corresponds *entity#1-entity#1* and KINSHIP relation, and to "*hand of the woman*" it corresponds *entity#1-entity#1* and PART-WHOLE relation. At this high level G^1, to each feature pair f_{ij}^1 it corresponds a number of semantic relations. For each feature, one can determine the most probable relation using equation (2). For instance, to feature *entity#1-entity#1* there correspond 13 relations and the most probable one is the PART-WHOLE relation as indicated by Table 2.

Step 2. Specialize the boundary

2.1 Constructing a lower boundary

This step consists of specializing the semantic classes of the ambiguous features. A feature f_{ij}^k on boundary G^k is ambiguous if it corresponds to more then one relation and its most relevant relation has a conditional probability less then 0.9. To eliminate non-important specializations, we specialize only the ambiguous classes that occurs in more than 1% of the training examples.

The specialization procedure consists of first identifying features f_{ij}^k to which correspond more than one semantic relation, then replace these features with their hyponyms synsets. Thus one feature breaks into several new specialized features. The net effect is that the semantic relations that were attached to f_{ij}^k will be "scattered" across the new specialized features. This process continues till each feature will have only one semantic relation attached. Each iteration creates a new boundary, as shown in Figure 1. Table 3 shows statistics of semantic features f_{ij}^k for each level of specialization G^k. Note the average number of relations per feature decreasing asymptotically to 1 as k increases.

2.2 Testing the new boundary

R	1	2	3	6	7	11	13	15	16	19	21	24	25	Others
$P(r\|entity - entity)$	0.048	0.120	0.006	0.032	0.430	0.016	0.035	0.285	0.012	0.004	0.010	0.001	0.001	0

Table 2: Sample row from the conditional probability table where the feature pair is *entity-entity*. The numbers in the top row identify the semantic relations (as in Table 1).

Boundary	Of-genitives			S-genitives		
	G^1	**G**^{2}	**G**^{3}	**G**^{1}	**G**^{2}	**G**^{3}
Number of modifi er features	9	31	74	9	37	91
Number head features	9	34	66	9	24	36
No. of feature pairs	63 out of 81	216 out of 1054	314 out of 4884	41 of 81	157 out of 888	247 out of 3276
Number of features with only one relation	26	153	281	14	99	200
Average number of relations per feature	3	1.46	1.14	3.59	1.78	1.36

Table 3: Statistics for the semantic class features by level of specialization.

The new boundary is more specific then the previous boundary and it is closer to the ideal boundary. However, we do not know how well it behaves on unseen examples and we are looking for a boundary that classifies with a high accuracy the unseen examples. We test the boundary on unseen examples. For that we used 10% of the annotated examples (different from the 10% of the examples used for testing) and compute the accuracy (f-measure) of the new boundary on them.

If the accuracy is larger than the previous boundary's accuracy, we are converging toward the best approximation of the boundary and thus we should repeat Step 2 for the new boundary.

If the accuracy is lower than the previous boundary's accuracy, the new boundary is too specific and the previous boundary is a better approximation of the ideal boundary.

For the automatic detection of the semantic relations encoded by genitives, the boundary constructed by the Semantic Scattering model is more apppropriate than a “tree cut”, like the ones used for verb disambiguation (McCarthy, 1997) (Li and Abe, 1998) and constructed using the Minimum Description Length model (Rissanen, 1978). The developement of a ”tree cut” model for the detection of the semantic relations encoded by genitives involves the construction of a different ”tree cut” for each head noun and threfore the usage of these cuts is restricted to those head nouns. On the other hand, Semantic Scattering constructs only one boundary that, unlike the ”tree cut” model, is general enough to classify any genitive construction, including the ones with constituents unseen during training.

4 Semantic Relations Classification Algorithm

The ideal boundary G^* is used for classifying the semantic relations encoded by genitives. The algorithm consists of:
Step 1. Process the sentence. Perform Word Sense Disambiguation and syntactic parsing of the sentence containing the genitive.
Step 2. Identify the head and modifier noun concepts.
Step 3. Identify the feature pair. Using the results from WSD and WordNet noun hierarchies, map the head and modifier concepts into the corresponding classes from the boundary and identify a feature pair f_{ij} that has the closest euclidean distance to the two classes.
Step 4. Find the semantic relation. Using the feature f_{ij}, determine the semantic relation that corresponds to that feature on the boundary. If there is no such relation, mark it as OTHER.

5 Results

For testing, we considered 20% of the annotated examples. We used half of the examples for detecting the boundary G^* and half for testing the system.

G^* Boundary Detection

The algorithm ran iteratively performing boundary

Results	Of-genitives			S-genitives		
	Baseline1	Baseline2	Results	Baseline1	Baseline2	Results
Number of correctly retrieved relations	49	59	**81**	15	27	**71**
Number of relations retrieved	73	75	**99**	63	66	**85**
Number of correct relations	104	104	**104**	96	96	**96**
Precision	67.12%	76.62%	**81.82%**	23.81%	40.91%	**83.53%**
Recall	47.12%	56.73%	**77.88%**	15.63%	28.13%	**73.96%**
F-measure	55.37%	65.92%	**79.80%**	18.87%	33.34%	**78.45%**

Table 4: Overall results for the semantic interpretation of genitives

specializations on the WordNet IS-A noun hierarchies in order to eliminate the ambiguities of the training examples. Boundary G^1 corresponds to the semantic classes of the nine WordNet noun hierarchies and boundaries G^2 and G^3 to their subsequent immediate hyponyms. For both s-genitives and of-genitives, boundary G^2 was more accurate then boundary G^1 and therefore we repeated Step 2. However, boundary G^3 was less accurate then boundary G^2 and thus boundary G^2 is the best approximation of the ideal boundary.

Semantic Relations Classification

Table 4 shows the results obtained when classifying the 36 relations (the 36th relation being OTHER) for of-genitives and s-genitives. The results are presented for the Semantic Scattering system that uses G^2 as the best approximation of the G^* together with two baselines. *Baseline1* system obtained the results without any word sense disambiguation (WSD) feature, i.e. using only the default sense number 1 for the concept pairs, and without any specialization. *Baseline2* system applied two iterations of the boundary detection algorithm but without any word sense disambiguation.

Overall, the Semantic Scattering System achieves an 81.82% precision and 77.88% recall for of-genitives and an 83.53% precision and 73.96% recall for s-genitives.

Both the WSD and the specialization are important for our system as indicated by the Baseline systems performance. The impact of specialization on the f-measure (Baseline2 minus Baseline1) is 10.55% for of-genitives and 14.47% for s-genitives, while the impact of WSD (final result minus Baseline2) is 14% for of-genitives and 45.11% for s-genitives.

Error Analysis

An important way of improving the performance of a system is to perform a detailed error analysis of the results. We have analyzed the various error sources encountered in our experiments and summarized the results in Table 5.

Error Type	Of-genitives %Error	S-genitives %Error
Missing feature	28.57	29.17
General semantic classes	28.57	20.83
WSD System	19.05	29.17
Reversed order of constituents	14.29	12.5
Named Entity Recognizer	4.76	8.33
Missing WordNet sense	4.76	0

Table 5: The error types encountered on the testing corpus.

6 Comparison with other Models

To evaluate our model, we have conducted experiments with other frequently used machine learning models, on the same dataset, using the same features. Table 6 shows a comparison between the results obtained with the Semantic Scattering algorithm and the decision trees (C5.0, http://www.rulequest.com/see5-info.html), the naive Bayes model (jBNC, Bayesian Network Classifier Toolbox, http://jbnc.sourceforge.net), and Support Vector Machine (libSVM, Chih-Chung Chang and Chih-Jen Lin. 2004. LIBSVM: a Library for Support Vector Machines, http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf). The reason for the superior performance of Semantic Scattering is because the classification of genitives is feature poor relying only on the semantics of the noun components, and the other three models normally work better with a larger set of features.

Accuracy	Of-genitives	S-genitives
Semantic Scattering	79.85%	78.75%
Decision Trees (C5.0)	40.60%	47.0%
Naive Bayes (JBNC)	42.31%	43.7%
SVM (LibSVM)	31.45 %	23.51%

Table 6: Accuracy performance of four learning models on the same testing corpus.

7 Discussion and Conclusions

The classification of genitives is an example of a learning problem where a tailored model outperforms other generally applicable models.

This paper presents a model for the semantic classification of genitives. A set of 35 semantic relations was identified, and we provided statistical evidence that when it comes to genitives, some relations are more frequent than others, while some are absent. The model relies on the semantic classes of noun constituents. The algorithm was trained and tested on 20,000 sentences containing 2,249 of-genitives and 1006 s-genitives and achieved an average precision of 82%, a recall of 76%, and an f-measure of 79%. For comparison, we ran a C5.0 learning system on the same corpus and obtained 40.60% accuracy for of-genitives and 47% for s-genitives. A similar experiment with a Naive Bayes learning system led to 42.31% accuracy for of-genitives and 43.7% for s-genitives. The performance with a Support Vector Machines learner was the worst, achieving only a 31.45% accuracy for of-genitives and 23.51% accuracy for s-genitives. We have also identified the sources of errors which when addressed may bring further improvements.

References

Barker, Chris. 1995. *Possessive Descriptions*. CSLI Publications, Standford, CA.

Berland, Matthew and Eugene Charniak. 1999. Finding parts in very large. In *Proceeding of ACL 1999*.

Fellbaum, Christiane. 1998. *WordNet - An Electronic Lexical Databases*. Cambridge MA: MIT Press.

Girju, Roxana, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the HLT-NAACL 2003*.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):277-295.

Hearst, Marti. 1998. Automated Discovery of WordNet relations. In *An Electronic Lexical Database and Some of its Applications*. MIT Press, Cambridge MA.

Lapata, Maria. 2000. Automatic Interpretation of Nominalizations. In *Proceedings of AAAI 2000*, 716-721.

Li, Hang and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics*, 24(2):217–224.

McCarthy, Diana. 1997. Word sense disambiguation for acquisition of selectional preferences. In *Proceedings of the ACL/EACL 97*.

Moldovan, Dan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the Semantic Classifi cation of Noun Phrases. In *Proceedings of the HLT-NAACL 2004, Computational Lexical Semantics Workshop*.

Nikiforidou, Kiki. 1991. The meanings of the genitive: A case study in the semantic structure and semantic change. *Cognitive Linguistics*, 2:149–205.

Novischi, Adrian, Dan Moldovan, Paul Parker, Adriana Badulescu, and Bob Hauser. 2004. *LCC's WSD systems for Senseval 3*. In *Proceedings of Senseval 3*.

Rissanen, Jorma. 1978. Modeling by shortest data description. *Automatic*, 14.

Rosario, Barbara and Marti Hearst. 2001. Classifying the Semantic Relations in Noun Compounds via a Domain-Specifi c Lexical Hierarchy. In *Proceeding of EMNLP 2001*.

Siegel, S. and N.J. Castellan. 1988. *Non Parametric Statistics for the behavioral sciences*. New York: McGraw-Hill.

Stefanowitsch, Anatol. 2001. Constructional semantics as a limit to grammatical alternation: Two genitives of English. *Determinants of Grammatical Variation in English*.

Taylor, John. 1996. *Possessives in English. An exploration in cognitive grammar*. Oxford, Clarendon Press.

Vikner, Carl and Per Anker Jensen. 1999. *A semantic analysis of the English genitive: interaction of lexical and formal semantics*.

Measuring the relative compositionality of verb-noun (V-N) collocations by integrating features

Sriram Venkatapathy[1]
Language Technologies Research Centre,
International Institute of Information
Technology - Hyderabad,
Hyderabad, India.
sriram@research.iiit.ac.in

Aravind K. Joshi
Department of Computer and
Information Science and Institute for
Research in Cognitive Science,
University of Pennsylvania,
Philadelphia, PA, USA.
joshi@linc.cis.upenn.edu

Abstract

Measuring the relative compositionality of Multi-word Expressions (MWEs) is crucial to Natural Language Processing. Various collocation based measures have been proposed to compute the relative compositionality of MWEs. In this paper, we define novel measures (both collocation based and context based measures) to measure the relative compositionality of MWEs of V-N type. We show that the correlation of these features with the human ranking is much superior to the correlation of the traditional features with the human ranking. We then integrate the proposed features and the traditional features using a SVM based ranking function to rank the collocations of V-N type based on their relative compositionality. We then show that the correlation between the ranks computed by the SVM based ranking function and human ranking is significantly better than the correlation between ranking of individual features and human ranking.

1 Introduction

The main goal of the work presented in this paper is to examine the relative compositionality of collocations of V-N type using a SVM based ranking function. Measuring the relative compositionality of V-N collocations is extremely helpful in applications such as machine translation where the collocations that are highly non-compositional can be handled in a special way (Schuler and Joshi, 2004) (Hwang and Sasaki, 2005).

Multi-word expressions (MWEs) are those whose structure and meaning cannot be derived from their component words, as they occur independently. Examples include conjunctions like 'as well as' (meaning 'including'), idioms like 'kick the bucket' (meaning 'die'), phrasal verbs like 'find out' (meaning 'search') and compounds like 'village community'. A typical natural language system assumes each word to be a lexical unit, but this assumption does not hold in case of MWEs (Becker, 1975) (Fillmore, 2003). They have idiosyncratic interpretations which cross word boundaries and hence are a 'pain in the neck' (Sag et al., 2002). They account for a large portion of the language used in day-to-day interactions (Schuler and Joshi, 2004) and so, handling them becomes an important task.

A large number of MWEs have a standard syntactic structure but are non-compositional semantically. An example of such a subset is the class of non-compositional verb-noun collocations (V-N collocations). The class of non-compositional V-N collocations is important because they are used very frequently. These include verbal idioms (Nunberg et al., 1994), support-verb constructions (Abeille, 1988), (Akimoto, 1989), among others. The expression 'take place' is a MWE whereas 'take a gift' is not a MWE.

[1]Part of the work was done at Institute for Research in Cognitive Science (IRCS), University of Pennsylvania, Philadelphia, PA 19104, USA, when he was visiting IRCS as a Visiting Scholar, February to December, 2004.

It is well known that one cannot really make a binary distinction between compositional and non-compositional MWEs. They do not fall cleanly into mutually exclusive classes, but populate the continuum between the two extremes (Bannard et al., 2003). So, we rate the MWEs (V-N collocations in this paper) on a scale from 1 to 6 where 6 denotes a completely compositional expression, while 1 denotes a completely opaque expression.

Various statistical measures have been suggested for ranking expressions based on their compositionality. Some of these are Frequency, Mutual Information (Church and Hanks, 1989) , distributed frequency of object (Tapanainen et al., 1998) and LSA model (Baldwin et al., 2003) (Schutze, 1998). In this paper, we define novel measures (both collocation based and context based measures) to measure the relative compositionality of MWEs of V-N type (see section 6 for details). Integrating these statistical measures should provide better evidence for ranking the expressions. We use a SVM based ranking function to integrate the features and rank the V-N collocations according to their compositionality. We then compare these ranks with the ranks provided by the human judge. A similar comparison between the ranks according to Latent-Semantic Analysis (LSA) based features and the ranks of human judges has been made by McCarthy, Keller and Caroll (McCarthy et al., 2003) for verb-particle constructions. (See Section 3 for more details). Some preliminary work on recognition of V-N collocations was presented in (Venkatapathy and Joshi, 2004).

We show that the measures which we have defined contribute greatly to measuring the relative compositionality of V-N collocations when compared to the traditional features. We also show that the ranks assigned by the SVM based ranking function correlated much better with the human judgement that the ranks assigned by individual statistical measures.

This paper is organized in the following sections **(1)** Basic Architecture, **(2)** Related work, **(3)** Data used for the experiments, **(4)** Agreement between the Judges, **(5)** Features, **(6)** SVM based ranking function, **(7)** Experiments & Results, and **(8)** Conclusion.

2 Basic Architecture

Every V-N collocation is represented as a vector of features which are composed largely of various statistical measures. The values of these features for the V-N collocations are extracted from the British National Corpus. For example, the V-N collocation 'raise an eyebrow' can be represented as
[Frequency = 271, Mutual Information = 8.43, Distributed frequency of object = 1456.29, etc.]. A SVM based ranking function uses these features to rank the V-N collocations based on their relative compositionality. These ranks are then compared with the human ranking.

3 Related Work

(Breidt, 1995) has evaluated the usefulness of the Point-wise Mutual Information measure (as suggested by (Church and Hanks, 1989)) for the extraction of V-N collocations from German text corpora. Several other measures like Log-Likelihood (Dunning, 1993), Pearson's χ^2 (Church et al., 1991), Z-Score (Church et al., 1991) , Cubic Association Ratio (MI3), etc., have been also proposed. These measures try to quantify the association of two words but do not talk about quantifying the non-compositionality of MWEs. Dekang Lin proposes a way to automatically identify the non-compositionality of MWEs (Lin, 1999). He suggests that a possible way to separate compositional phrases from non-compositional ones is to check the existence and mutual-information values of phrases obtained by replacing one of the words with a similar word. According to Lin, a phrase is probably non-compositional if such substitutions are not found in the collocations database or their mutual information values are significantly different from that of the phrase. Another way of determining the non-compositionality of V-N collocations is by using 'distributed frequency of object' (DFO) in V-N collocations (Tapanainen et al., 1998). The basic idea in there is that "if an object appears only with one verb (or few verbs) in a large corpus we expect that it has an idiomatic nature" (Tapanainen et al., 1998).

Schone and Jurafsky (Schone and Jurafsky, 2001) applied Latent-Semantic Analysis (LSA) to the analysis of MWEs in the task of MWE discovery, by way of rescoring MWEs extracted from the corpus. An interesting way of quantifying the relative compositionality of a MWE is proposed by Baldwin, Bannard, Tanaka and Widdows (Baldwin et al., 2003). They use LSA to determine the similarity between an MWE and its constituent words, and claim that higher similarity indicates great decomposability. In terms of compositionality, an expression is likely to be relatively more compositional if it is decomposable. They evaluate their model on English NN compounds and verb-particles, and showed that the model correlated moderately well with the Wordnet based decomposability theory (Baldwin et al., 2003).

McCarthy, Keller and Caroll (McCarthy et al., 2003) judge compositionality according to the degree of overlap in the set of most similar words to the verb-particle and head verb. They showed that the correlation between their measures and the human ranking was better than the correlation between the statistical features and the human ranking. We have done similar experiments in this paper where we compare the correlation value of the ranks provided by the SVM based ranking function with the ranks of the individual features for the V-N collocations. We show that the ranks given by the SVM based ranking function which integrates all the features provides a significantly better correlation than the individual features.

4 Data used for the experiments

The data used for the experiments is British National Corpus of 81 million words. The corpus is parsed using Bikel's parser (Bikel, 2004) and the Verb-Object Collocations are extracted. There are 4,775,697 V-N collocations of which 1.2 million are unique. All the V-N collocations above the frequency of 100 (n=4405) are taken to conduct the experiments so that the evaluation of the system is feasible. These 4405 V-N collocations were searched in Wordnet, American Heritage Dictionary and SAID dictionary (LDC,2003). Around 400 were found in at least one of the dictionaries. Another 400 were extracted from the rest so that the evaluation set has roughly equal number of compositional and non-compositional expressions. These 800 expressions were annotated with a rating from 1 to 6 by using guidelines independently developed by the authors. 1 denotes the expressions which are totally non-compositional while 6 denotes the expressions which are totally compositional. The brief explanation of the various ratings is as follows: **(1)** No word in the expression has any relation to the actual meaning of the expression. Example : "**leave a mark**". **(2)** Can be replaced by a single verb. Example : "**take a look**". **(3)** Although meanings of both words are involved, at least one of the words is not used in the usual sense. Example : "**break news**". **(4)** Relatively more compositional than (3). Example : "**prove a point**". **(5)** Relatively less compositional than (6). Example : "**feel safe**". **(6)** Completely compositional. Example : "**drink coffee**".

5 Agreement between the Judges

The data was annotated by two fluent speakers of English. For 765 collocations out of 800, both the annotators gave a rating. For the rest, at least one of the annotators marked the collocations as "don't know". Table 1 illustrates the details of the annotations provided by the two judges.

Ratings	6	5	4	3	2	1
Annotator1	141	122	127	119	161	95
Annotator2	303	88	79	101	118	76

Table 1: Details of the annotations of the two annotators

From the table 1 we see that annotator1 distributed the rating more uniformly among all the collocations while annotator2 observed that a significant proportion of the collocations were completely compositional. To measure the agreement between the two annotators, we used the Kendall's TAU (τ) (Siegel and Castellan, 1988). τ is the correlation between the rankings[1] of collocations given by the two annotators. τ ranges between 0 (little agreement) and 1 (full agreement). τ is defined as,

$$\tau = \frac{\sum_{i<j} sgn(x_i - x_j)\ sgn(y_i - y_j)}{\sqrt{(t_0 - t_1)\ (t_0 - t_2)}}$$

$$t_1 = \frac{\sum u_i\ (u_i - 1)}{2}\ ,\ t_2 = \frac{\sum v_i\ (v_i - 1)}{2}\ ,\ t_0 = \frac{n\ (n-1)}{2}$$

[1]computed from the ratings

where x_i's are the rankings of annotator1 and y_i's are the rankings of annotator2, n is the number of collocations, u_i is the number of values in the i^{th} group of tied x values and v_i is the number of values in the i^{th} group of tied y values.

We obtained a τ score of **0.61** which is highly significant. This shows that the annotators were in a good agreement with each other in deciding the rating to be given to the collocations. We also compare the ranking of the two annotators using Pearson's Rank-Correlation coefficient (r_s) (Siegel and Castellan, 1988). We obtained a r_s score of **0.71** indicating a good agreement between the annotators. A couple of examples where the annotators differed are **(1)** "perform a task" was rated 3 by annotator1 while it was rated 6 by annotator2 and **(2)** "pay tribute" was rated 1 by annotator1 while it was rated 4 by annotator2.

The 765 samples annotated by both the annotators were then divided into a training set and a testing set in several possible ways to cross-validate the results of ranking (section 8).

6 Features

Each collocation is represented by a vector whose dimensions are the statistical features obtained from the British National Corpus. The features used in our experiments can be classified as (1) Collocation based features and (2) Context based features.

6.1 Collocation based features

Collocation based features consider the entire collocation as an unit and compute the statistical properties associated with it. The collocation based features that we considered in our experiments are (1) Frequency, (2) Point-wise Mutual Information, (3) Least mutual information difference with similar collocations, (4) Distributed frequency of object and (5) Distributed frequency of object using the verb information.

6.1.1 Frequency (f)

This feature denotes the frequency of a collocation in the British National Corpus. Cohesive expressions have a high frequency. Hence, greater the frequency, the more is the likelihood of the expression to be a MWE.

6.1.2 Point-wise Mutual Information (m)

Point-wise Mutual information of a collocation (Church and Hanks, 1989) is defined as,

$$m(v,o) = \frac{f(v,o) * f(-,-)}{f(v,-) * f(-,o)}$$

where, v is the verb and o is the object of the collocation. The higher the Mutual information of a collocation, the more is the likelihood of the expression to be a MWE.

6.1.3 Least mutual information difference with similar collocations (μ)

This feature is based on Lin's work (Lin, 1999). He suggests that a possible way to separate compositional phrases from non-compositional ones is to check the existence and mutual information values of *similar collocations* (phrases obtained by replacing one of the words with a similar word). For example, 'eat apple' is a similar collocation of 'eat pear'.

For a collocation, we find the similar collocations by substituting the verb and the object with their *similar words*[2]. The similar collocation having the least mutual information difference is chosen and the difference in their mutual information values is noted.

If a collocation c has a set of similar collocations S, then we define μ as

$$\mu(v_c, o_c) = min_{s \in S} \ [\ abs \ (m(c) - m(s)) \ ]$$

where $abs(x)$ returns the absolute value of x and v_c and o_c are the verb and object of the collocation c respectively. If similar collocations do not exist for a collocation, then this feature is assigned the highest among the values assigned in the previous equation. In this case, μ is defined as,

$$\mu(v,o) = max_{i,j} \ [\ \mu(v_i, o_j) \ ]$$

where v and o are the verb and object of collocations for which similar collocations do not exist. The higher the value of μ, the more is the likelihood of the collocation to be a MWE.

[2]obtained from Lin's (Lin, 1998) automatically generated thesaurus (http://www.cs.ualberta.ca/∼lindek/downloads.htm). We obtained the best results (section 8) when we substituted top-5 similar words for both the verb and the object. To measure the compositionality, semantically similar words are more suitable than synomys. Hence, we choose to use Lin's thesaurus (Lin, 1998) instead of Wordnet (Miller et al., 1990).

6.1.4 Distributed Frequency of Object (d)

The distributed frequency of object is based on the idea that "if an object appears only with one verb (or few verbs) in a large corpus, the collocation is expected to have idiomatic nature" (Tapanainen et al., 1998). For example, 'sure' in 'make sure' occurs with very few verbs. Hence, 'sure' as an object is likely to give a special sense to the collocation as it cannot be used with any verb in general. It is defined as,

$$d(o) = \frac{\sum_i^n f(v_i, o)}{n}$$

where n is the number of verbs occurring with the object (o), v_i's are the verbs cooccuring with o and $f(v_i, o) > T$. As the number of verbs (n) increases, the value of $d(o)$ decreases. Here, T is a threshold which can be set based on the corpus. This feature treats 'point finger' and 'polish finger' in the same way as it does not use the information specific to the verb in the collocation. Here, both the collocations will have the value $d(``finger'')$. The 3 collocations having the highest value of this feature are (1) come true, (2) become difficult and (3) make sure.

6.1.5 Distributed Frequency of Object using the Verb information (ρ)

Here, we have introduced an extension to the feature d such that the collocations like 'point finger' and 'polish finger' are treated differently and more appropriately. This feature is based on the idea that "a collocation is likely to be idiomatic in nature if there are only few other collocations with the same object and dissimilar verbs". We define this feature as,

$$\rho(v, o) = \frac{\sum_i^n f(v_i, o) \ * \ dist(v, v_i)}{n}$$

where $n < 50$ is the number of verbs occurring with o, v_i's are the verbs cooccuring with o and $f(v_i, o) > T$. $dist(v, v_i)$ is the distance between the verb v and v_i. It is calculated using the wordnet similarity measure defined by Hirst and Onge (Hirst and St-Onge, 1998). In our experiments, we considered top-50 verbs which co-occurred with the object o. We used a Perl package Wordnet::Similarity by Patwardhan[3] to conduct our experiments.

[3]http://www.d.umn.edu/~tpederse/similarity.html

6.2 Context based features

Context based measures use the context of a word/collocation to measure their properties. We represented the context of a word/collocation using a LSA model. LSA is a method of representing words/collocations as points in vector space.

The LSA model we built is similar to that described in (Schutze, 1998) and (Baldwin et al., 2003). First, 1000 most frequent content words (i.e., not in the stop-list) were chosen as "content-bearing words". Using these content-bearing words as column labels, the 50,000 most frequent terms in the corpus were assigned row vectors by counting the number of times they occurred within the same sentence as content-bearing words. Principal component analysis was used to determine the principal axis and we get the transformation matrix $A_{1000X100}$ which can be used to reduce the dimensions of the 1000 dimensional vectors to 100 dimensions.

We will now describe in Sections 6.2.1 and 6.2.2 the features defined using LSA model.

6.2.1 Dissimilarity of the collocation with its constituent verb using the LSA model (ι)

If a collocation is highly dissimilar to its constituent verb, it implies that the usage of the verb in the specific collocation is not in a general sense. For example, the sense of 'change' in 'change hands' would be very different from its usual sense. Hence, the greater the dissimilarity between the collocation and its constituent verb, the more is the likelihood that it is a MWE. The feature is defined as

$$\iota(c, v_c) = 1 - sim_{LSA}(c, v_c)$$

$$sim_{LSA}(c, v_c) = \frac{lsa(c) \ . \ lsa(v_c)}{|lsa(c)| * |lsa(v_c)|}$$

where, c is the collocation, v_c is the verb of the collocation and lsa(x) is representation of x using the LSA model.

6.2.2 Similarity of the collocation to the verb-form of the object using the LSA model (ψ)

If a collocation is highly similar to the verb form of an object, it implies that the verb in the collocation does not contribute much to the meaning of the collocation. The verb either acts as a sort of support verb, providing perhaps some additional aspectual meaning. For example, the verb 'give' in 'give a smile' acts merely as a support verb. Here, the collocation 'give a smile' means the same as the verb-form of the object i.e., 'to smile'. Hence, the greater is the similarity between the collocation and the verb-form of the object, the more is the likelihood that it is a MWE. This feature is defined as

$$\psi(c, vo_c) = \frac{lsa(c)\ .\ lsa(vo_c)}{|lsa(c)| * |lsa(vo_c)|}$$

where, c is the collocation and vo_c is the verb-form of the object o_c. We obtained the verb-form of the object from the wordnet (Miller et al., 1990) using its 'Derived forms'. If the object doesn't have a verbal form, the value of this feature is 0. Table 2 contains the top-6 collocations according to this feature. All the collocations in Table 2 (except 'receive award' which does not mean the same as 'to award') are good examples of MWEs.

Collocation	Value		Collocation	Value
pay visit	0.94		provide assistance	0.92
provide support	0.93		give smile	0.92
receive award	0.92		fi nd solution	0.92

Table 2: Top-6 collocations according to this feature

7 SVM based ranking function/algorithm

The optimal rankings on the training data is computed using the average ratings of the two users. The goal of the learning function is to model itself according to this rankings. It should take a ranking function f from a family of ranking functions F that maximizes the empirical τ (Kendall's Tau). τ expresses the similarity between the optimal ranking (r^*) and the ranking (r_f) computed by the function f. SVM-Light[4] is a tool developed by Joachims (Joachims, 2002) which provides us such a function. We briefly describe the algorithm in this section.

Maximizing τ is equivalent to minimizing the number of discordant pairs (the pairs of collocations which are not in the same order as in the optimal ranking). This is equivalent to finding the weight vector $\bar{w}$ so that the maximum number of inequalities are fulfilled.

$$\forall (c_i, c_j) \in r^* : \bar{w}\ \phi(c_i) > \bar{w}\ \phi(c_j)$$

where c_i and c_j are the collocations, $(c_i, c_j) \in r^*$ if the collocation c_i is ranked higher than c_j for the optimal ranking r^*, $\phi(c_i)$ and $\phi(c_j)$ are the mapping onto features (section 6) that represent the properties of the V-N collocations c_i and c_j respectively and $\bar{w}$ is the weight vector representing the ranking function f_w.

Adding SVM regularization for margin maximization to the objective leads to the following optimization problem (Joachims, 2002).

$$minimize: \quad V(\bar{w}, \bar{\epsilon}) = \frac{1}{2}\, \bar{w}\bar{w} + C \sum \epsilon_{i,j}$$

$$constr: \bar{w}(\phi(c_i) - \phi(c_j)) \geq 1 - \epsilon_{i,j},\ \forall_{i,j}\ \epsilon_{i,j} \geq 0$$

where $\epsilon_{i,j}$ are the (non-negative) slack variables and C is the margin that allows trading-off margin size against training error. This optimization problem is equivalent to that of a classification SVM on pairwise difference vectors $\phi(c_i)$ - $\phi(c_j)$. Due to similarity, it can be solved using decomposition algorithms similar to those used for SVM classification (Joachims, 1999).

Using the learnt function $f_{\bar{w}^*}$ ($\bar{w}^*$ is the learnt weight vector), the collocations in the test set can be ranked by computing their values using the formula below.

$$rc(c_i) = \bar{w}^* \phi(c_i)$$

8 Experiments and Results

For training, we used 10% of the data and for testing, we use 90% of the data as the goal is to use only a small portion of the data for training (Data was divided in 10 different ways for cross-validation. The results presented here are the average results).

All the statistical measures show that the expressions ranked higher according to their decreasing values are more likely to be non-compositional. We compare these ranks with the human rankings (obtained using the average ratings of the users). To compare, we use Pearson's Rank-Order Correlation Coefficient (r_s) (Siegel and Castellan, 1988).

We integrate all the seven features using the SVM based ranking function (described in section 7). We

[4]http://svmlight.joachims.org

see that the correlation between the relative compositionality of the V-N collocations computed by the SVM based ranking function is significantly higher than the correlation between the individual features and the human ranking (Table 3).

Feature	Correlation		Feature	Correlation
f (f1)	0.129		ρ (f5)	0.203
m (f2)	0.117		ι (f6)	0.139
μ (f3)	0.210		ψ (f7)	*0.300*
d (f4)	0.111		Ranking f	**0.448**

Table 3: The correlation values of the ranking of individual features and the ranking of SVM based ranking function with the ranking of human judgements

In table 3, we also see that the contextual feature which we proposed, 'Similarity of the collocation to the verb-form of the object' (ψ), correlated significantly higher than the other features which indicates that it is a good measure to represent the semantic compositionality of V-N expressions. Other expressions which were good indicators when compared to the traditional features are 'Least mutual information difference with similar collocations' (μ) and 'Distributed frequency of object using the verb information' (ρ).

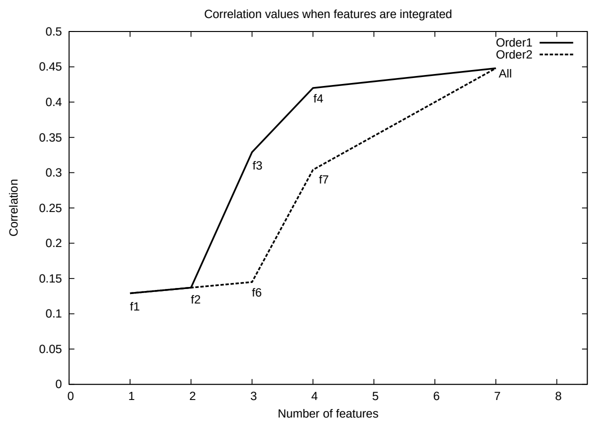


Figure 1: The change in r_s as more features are added to the ranking function

To observe the contribution of the features to the SVM based ranking function, we integrate the features (section 6) one after another (in two different ways) and compute the relative order of the collocations according to their compositionality. We see that as we integrate more number of relevant compositionality based features, the relative order correlates better (better r_s value) with the human ranking (Figure 1). We also see that when the feature 'Least mutual information difference with similar collocations' is added to the SVM based ranking function, there is a high rise in the correlation value indicating it's relevance. In figure 1, we also observe that the context-based features did not contribute much to the SVM based ranking function even though they performed well individually.

9 Conclusion

In this paper, we proposed some collocation based and contextual features to measure the relative compositionality of MWEs of V-N type. We then integrate the proposed features and the traditional features using a SVM based ranking function to rank the V-N collocations based on their relative compositionality. Our main results are as follows, **(1)** The properties 'Similarity of the collocation to the verb-form of the object', ' Least mutual information difference with similar collocations' and 'Distributed frequency of object using the verb information' contribute greatly to measuring the relative compositionality of V-N collocations. **(2)** The correlation between the ranks computed by the SVM based ranking function and the human ranking is significantly better than the correlation between ranking of individual features and human ranking.

In future, we will evaluate the effectiveness of the techniques developed in this paper for applications like Machine Translation. We will also extend our approach to other types of MWEs and to the MWEs of other languages (work on Hindi is in progress).

Acknowledgments

We want to thank the anonymous reviewers for their extremely useful reviews. We are grateful to Roderick Saxey and Pranesh Bhargava for annotating the data which we used in our experiments.

References

Anne Abeille. 1988. Light verb constructions and extraction out of np in a tree adjoining grammar. In *Pa-*

pers of the 24th Regional Meeting of the Chicago Linguistics Society.

Monoji Akimoto. 1989. Papers of the 24th regional meeting of the chicago linguistics society. In *Shinozaki Shorin*.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*.

Joseph D. Becker. 1975. The phrasal lexicon. In *Theoritical Issues of NLP, Workshop in CL, Linguistics, Psychology and AI, Cambridge, MA*.

Daniel M. Bikel. 2004. A distributional analysis of a lexicalized statistical parsing model. In *Proceedings of EMNLP*.

Elisabeth Breidt. 1995. Extraction of v-n-collocations from text corpora: A feasibility study for german. In *CoRR-1996*.

K. 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, 1990*.

K. Church, W. Gale, P. Hanks, and D. Hindle. 1991. Parsing, word associations and typical predicate-argument relations. In *Current Issues in Parsing Technology. Kluwer Academic, Dordrecht, Netherlands, 1991*.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. In *Computational Linguistics - 1993*.

Charles Fillmore. 2003. An extremist approach to multi-word expressions. In *A talk given at IRCS, University of Pennsylvania, 2003*.

G. Hirst and D. St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In *Fellbaum C., ed., Wordnet: An electronic lexical database. MIT Press*.

Young-Sook Hwang and Yutaka Sasaki. 2005. Context-dependent SMT model using bilingual verb-noun collocation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

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

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL'98*.

Dekang Lin. 1999. Automatic identifi cation of non-compositional phrases. In *Proceedings of ACL-99, College Park, USA*.

D. McCarthy, B. Keller, and J. Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment, 2003*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to wordnet: an on-line lexical database. In *International Journal of Lexicography*.

G. Nunberg, I. A. Sag, and T. Wasow. 1994. Idioms. In *Language, 1994*.

I. A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multi-word expressions: a pain in the neck for nlp. In *Proceedings of CICLing , 2002*.

Patrick Schone and Dan Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of EMNLP , 2001*.

William Schuler and Aravind K. Joshi. 2004. Relevance of tree rewriting systems for multi-word expressions. In *To be published*.

Hinrich Schutze. 1998. Automatic word-sense discrimination. In *Computational Linguistics*.

S. Siegel and N. John Castellan. 1988. In *Non-parametric Statistics of the Behavioral Sciences*. McGraw-Hill, NJ.

Pasi Tapanainen, Jussi Piitulaine, and Timo Jarvinen. 1998. Idiomatic object usage and support verbs. In *36th Annual Meeting of the Association for Computational Linguistics*.

Sriram Venkatapathy and Aravind K. Joshi. 2004. Recognition of multi-word expressions: A study of verb-noun (v-n) collocations. In *Proceedings of the International Conference on Natural Language Processing,2004*.

A Semi-Supervised Feature Clustering Algorithm with Application to Word Sense Disambiguation

Zheng-Yu Niu, Dong-Hong Ji
Institute for Infocomm Research
21 Heng Mui Keng Terrace
119613 Singapore
{zniu, dhji}@i2r.a-star.edu.sg

Chew Lim Tan
Department of Computer Science
National University of Singapore
3 Science Drive 2
117543 Singapore
tancl@comp.nus.edu.sg

Abstract

In this paper we investigate an application of feature clustering for word sense disambiguation, and propose a semi-supervised feature clustering algorithm. Compared with other feature clustering methods (ex. supervised feature clustering), it can infer the distribution of class labels over (unseen) features unavailable in training data (labeled data) by the use of the distribution of class labels over (seen) features available in training data. Thus, it can deal with both seen and unseen features in feature clustering process. Our experimental results show that feature clustering can aggressively reduce the dimensionality of feature space, while still maintaining state of the art sense disambiguation accuracy. Furthermore, when combined with a semi-supervised WSD algorithm, semi-supervised feature clustering outperforms other dimensionality reduction techniques, which indicates that using unlabeled data in learning process helps to improve the performance of feature clustering and sense disambiguation.

1 Introduction

This paper deals with word sense disambiguation (WSD) problem, which is to assign an appropriate sense to an occurrence of a word in a given context. Many corpus based statistical methods have been proposed to solve this problem, including supervised learning algorithms (Leacock et al., 1998; Towel and Voorheest, 1998), weakly supervised learning algorithms (Dagan and Itai, 1994; Li and Li, 2004; Mihalcea, 2004; Niu et al., 2005; Park et al., 2000; Yarowsky, 1995), unsupervised learning algorithms (or word sense discrimination) (Pedersen and Bruce, 1997; Sch$\ddot{u}$tze, 1998), and knowledge based algorithms (Lesk, 1986; McCarthy et al., 2004).

In general, the most common approaches start by evaluating the co-occurrence matrix of features versus contexts of instances of ambiguous word, given sense-tagged training data for this target word. As a result, contexts are usually represented in a high-dimensional sparse feature space, which is far from optimal for many classification algorithms. Furthermore, processing data lying in high-dimensional feature space requires large amount of memory and CPU time, which limits the scalability of WSD model to very large datasets or incorporation of WSD model into natural language processing systems.

Standard dimentionality reduction techniques include (1) supervised feature selection and supervised feature clustering when given labeled data, (2) unsupervised feature selection, latent semantic indexing, and unsupervised feature clustering when only unlabeled data is available. Supervised feature selection improves the performance of an examplar based learning algorithm over SENSEVAL-2 data (Mihalcea, 2002), Naive Bayes and decision tree over SENSEVAL-1 and SENSEVAL-2 data (Lee and Ng, 2002), but feature selection does not improve SVM and Adaboost over SENSEVAL-1 and SENSEVAL-2 data (Lee and Ng, 2002) for word sense disambiguation. Latent semantic indexing (LSI) studied in (Sch$\ddot{u}$tze, 1998) improves the performance of sense discrimination, while unsupervised feature selection also improves the performance of word sense discrimination (Niu et al., 2004). But little work is done on using feature clustering to conduct dimensionality reduction for WSD. This paper will describe an application of feature clustering technique to WSD task.

Feature clustering has been extensively studied for the benefit of text categorization and document clustering. In the context of text categorization, supervised feature clustering algorithms (Baker and McCallum, 1998; Bekkerman et al., 2003; Slonim and Tishby, 2001) usually cluster words into groups based on the distribution of class labels over features, which can compress the feature space much more aggressively while still maintaining state of the art classification accuracy. In the context of document clustering, unsupervised feature clustering algorithms (Dhillon, 2001; Dhillon et al., 2002; Dhillon et al., 2003; El-Yaniv and Souroujon, 2001; Slonim and Tishby, 2000) perform word clustering by the use of word-document co-occurrence matrix, which can improve the performance of document clustering by clustering documents over word clusters.

Supervised feature clustering algorithm groups features into clusters based on the distribution of class labels over features. But it can not group unseen features (features that do not occur in labeled data) into meaningful clusters since there are no class labels associated with these unseen features. On the other hand, while given labeled data, unsupervised feature clustering method can not utilize class label information to guide feature clustering procedure. While, as a promising classification strategy, semi-supervised learning methods (Zhou et al., 2003; Zhu and Ghahramani, 2002; Zhu et al., 2003) usually utilize all the features occurring in labeled data and unlabeled data. So in this paper we propose a semi-supervised feature clustering algorithm to overcome this problem. Firstly, we try to induce class labels for unseen features based on the similarity among seen features and unseen features. Then all the features (including seen features and unseen features) are clustered based on the distribution of class labels over them.

This paper is organized as follows. First, we will formulate a feature clustering based WSD problem in section 2. Then in section 3 we will describe a semi-supervised feature clustering algorithm. Section 4 will provide experimental results of various dimensionality reduction techniques with combination of state of the art WSD algorithms on SENSEVAL-3 data. Section 5 will provide a review of related work on feature clustering. Finally we will conclude our work and suggest possible improvement in section 6.

2 Problem Setup

Let $X = \{x_i\}_{i=1}^{n}$ be a set of contexts of occurrences of an ambiguous word w, where x_i represents the context of the i-th occurrence, and n is the total number of this word's occurrences. Let $S = \{s_j\}_{j=1}^{c}$ denote the sense tag set of w. The first l examples $x_g (1 \leq g \leq l)$ are labeled as y_g ($y_g \in S$) and other u ($l+u = n$) examples $x_h (l+1 \leq h \leq n)$ are unlabeled. The goal is to predict the sense of w in context x_h by the use of label information of x_g and similarity information among examples in X.

We use $\widetilde{F}$ to represent feature clustering result into $N_{\widetilde{F}}$ clusters when F is a set of features. After feature clustering, any context x_i in X can be represented as a vector over feature clusters $\widetilde{F}$. Then we can use supervised methods (ex. SVM) (Lee and Ng, 2002) or semi-supervised methods (ex. label propagation algorithm) (Niu et al., 2005) to perform sense disambiguation on unlabeled instances of target word.

3 Semi-Supervised Feature Clustering Algorithm

In supervised feature clustering process, F consists of features occurring in the first l labeled examples, which can be denoted as F_L. But in the setting of transductive learning, semi-supervised learning algorithms will utilize not only the features in labeled examples (F_L), but also unseen features in unlabeled examples (denoted as $F_{\overline{L}}$). $F_{\overline{L}}$ consists of the features that occur in unlabeled data, but never appear in labeled data.

Supervised feature clustering algorithm usually performs clustering analysis over feature-class matrix, where each entry (i, j) in this matrix is the number of times of the i-th feature co-occurring with the j-th class. Therefore it can not group features in $F_{\overline{L}}$ into meaningful clusters since there are no class labels associated with these features. We overcome this problem by firstly inducing class labels for unseen features based on the similarity among features in F_L and $F_{\overline{L}}$, then clustering all the features (including F_L and $F_{\overline{L}}$) based on the distribution of class labels over them.

This semi-supervised feature clustering algorithm is defined as follows:

Input:

Feature set $F = F_L \bigcup F_{\overline{L}}$ (the first $|F_L|$ features in F belong to F_L, and the remaining $|F_{\overline{L}}|$ features belong to $F_{\overline{L}}$), context set X, the label information of $x_g (1 \leq g \leq l)$, $N_{\widetilde{F}}$ (the number of clusters in $\widetilde{F}$);

Output:

Clustering solution $\widetilde{F}$;

Algorithm:

1. Construct $|F| \times |X|$ feature-example matrix $M^{F,X}$, where entry $M^{F,X}_{i,j}$ is the number of times of f_i co-occurring with example x_j ($1 \leq j \leq n$).

2. Form $|F| \times |F|$ affinity matrix W defined by $W_{ij} = exp(-\frac{d^2_{ij}}{\sigma^2})$ if $i \neq j$ and $W_{ii} = 0$ ($1 \leq i, j \leq |F|$), where d_{ij} is the distance (ex. Euclidean distance) between f_i (the i-th row in $M^{F,X}$) and f_j (the j-th row in $M^{F,X}$), and σ is used to control the weight W_{ij}.

3. Construct $|F_L| \times |S|$ feature-class matrix $Y^{F_L,S}$, where the entry $Y^{F_L,S}_{i,j}$ is the number of times of feature f_i ($f_i \in F_L$) co-occurring with sense s_j.

4. Obtain hard label matrix for features in F_L (denoted as $Y^{F_L,S}_{hard}$) based on $Y^{F_L,S}$, where entry $Y^{F,S}_{hard\ i,j} = 1$ if the hard label of f_i is s_j, otherwise zero. Obtain hard labels for features in $F_{\overline{L}}$ using a classifier based on W and $Y^{F_L,S}_{hard}$. In this paper we use label propagation (LP) algorithm (Zhu and Ghahramani, 2002) to get hard labels for $F_{\overline{L}}$.

5. Construct $|F| \times |S|$ feature-class matrix $Y^{F,S}_{hard}$, where entry $Y^{F,S}_{hard\ i,j} = 1$ if the hard label of f_i is s_j, otherwise zero.

6. Construct the matrix $L = D^{-1/2} W D^{-1/2}$ in which D is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of W.

7. Label each feature in F as soft label $\widehat{Y}^{F,S}_i$, the i-th row of $\widehat{Y}^{F,S}$, where $\widehat{Y}^{F,S} = (I - \alpha L)^{-1} Y^{F,S}_{hard}$.

8. Obtain the feature clustering solution $\widetilde{F}$ by clustering the rows of $\widehat{Y}^{F,S}_i$ into $N_{\widetilde{F}}$ groups. In this paper we use sequential information bottleneck (sIB) algorithm (Slonim and Tishby, 2000) to perform clustering analysis.

End

Step $3 \sim 5$ are the process to obtain hard labels for features in F, while the operation in step 6 and 7 is a local and global consistency based semi-supervised learning (LGC) algorithm (Zhou et al., 2003) that smooth the classification result of LP algorithm to acquire a soft label for each feature.

At first sight, this semi-supervised feature clustering algorithm seems to make little sense. Since we run feature clustering in step 8, why not use LP algorithm to obtain soft label matrix $Y^{F_{\overline{L}},S}$ for features in $F_{\overline{L}}$ by the use of $Y^{F_L,S}$ and W, then just apply sIB directly to soft label matrix $\widehat{Y}^{F,S}$ (constructed by catenating $Y^{F_L,S}$ and $Y^{F_{\overline{L}},S}$)?

The reason for using LGC algorithm to acquire soft labels for features in F is that in the context of transductive learning, the size of labeled data is rather small, which is much less than that of unlabeled data. This makes it difficult to obtain reliable estimation of class label's distribution over features from only labeled data. This motivates us to use raw information (hard labels of features in F_L) from labeled data to estimate hard labels of features in $F_{\overline{L}}$. Then LGC algorithm is used to smooth the classification result of LP algorithm based on the assumption that a good classification should change slowly on the coherent structure aggregated by a large amount of unlabeled data. This operation makes our algorithm more robust to the noise in feature-class matrix $Y^{F_L,S}$ that is estimated from labeled data.

In this paper, σ is set as the average distance between labeled examples from different classes, and $N_{\widetilde{F}} = |F|/10$. Latent semantic indexing technique (LSI) is used to perform factor analysis in $M^{F,X}$ before calculating the distance between features in step 2.

4 Experiments and Results

4.1 Experiment Design

For empirical study of dimensionality reduction techniques on WSD task, we evaluated five dimensionality reduction algorithms on the data in English lexical sample (ELS) task of SENSEVAL-3 (Mihalcea et al., 2004)(including all the 57 English words) [1]: supervised feature clustering (SuFC) (Baker and McCallum, 1998; Bekkerman et al., 2003; Slonim

[1]Available at http://www.senseval.org/senseval3

and Tishby, 2001), iterative double clustering (IDC) (El-Yaniv and Souroujon, 2001), semi-supervised feature clustering (SemiFC) (our algorithm), supervised feature selection (SuFS) (Forman, 2003), and latent semantic indexing (LSI) (Deerwester et. al., 1990) [2].

We used sIB algorithm [3] to cluster features in F_L into groups based on the distribution of class labels associated with each feature. This procedure can be considered as our re-implementation of supervised feature clustering. After feature clustering, examples can be represented as vectors over feature clusters.

IDC is an extension of double clustering method (DC) (Slonim and Tishby, 2000), which performs iterations of DC. In the transductive version of IDC, they cluster features in F as distributions over class labels (given by the labeled data) during the first stage of the IDC first iteration. This phase results in feature clusters $\widetilde{F}$. Then they continue as usual; that is, in the second phase of the first IDC iteration they group X into $N_{\widetilde{X}}$ clusters, where X is represented as distribution over $\widetilde{F}$. Subsequent IDC iterations use all the unlabeled data. This IDC algorithm can result in two clustering solutions: $\widetilde{F}$ and $\widetilde{X}$. Following (El-Yaniv and Souroujon, 2001), the number of iterations is set as 15, and $N_{\widetilde{X}} = |S|$ (the number of senses of target word) in our re-implementation of IDC. After performing IDC, examples can be represented as vectors over feature clusters $\widetilde{F}$.

Supervised feature selection has been extensively studied for text categorization task (Forman, 2003). Information gain (IG) is one of state of the art criteria for feature selection, which measures the decrease in entropy when the feature is given vs. absent. In this paper, we calculate IG score for each feature in F_L, then select top $|F|/10$ features with highest scores to form reduced feature set. Then examples can be represented as vectors over the reduced feature set.

LSI is an unsupervised factor analysis technique based on Singular Value Decomposition of a $|X| \times |F|$ example-feature matrix. The underlying technique for LSI is to find an orthogonal basis for the feature-example space for which the axes lie along the dimensions of maximum variance. After using LSI on the example-feature matrix, we can get vector representation for each example in X in reduced feature space.

For each ambiguous word in ELS task of SENSEVAL-3, we used three types of features to capture contextual information: part-of-speech of neighboring words with position information, unordered single words in topical context, and local collocations (as same as the feature set used in (Lee and Ng, 2002) except that we did not use syntactic relations). We removed the features with occurrence frequency (counted in both training set and test set) less than 3 times.

We ran these five algorithms for each ambiguous word to reduce the dimensionality of feature space from $|F|$ to $|F|/10$ no matter which training data is used (ex. full SENSEVAL-3 training data or sampled SENSEVAL-3 training data). Then we can obtain new vector representation of X in new feature space acquired by SuFC, IDC, SemiFC, and LSI or reduced feature set by SuFS.

Then we used SVM [4] and LP algorithm to perform sense disambiguation on vectors in dimensionality reduced feature space. SVM and LP were evaluated using accuracy [5] (fine-grained score) on test set of SENSEVAL-3. For LP algorithm, the test set in SENSEVAL-3 data was also used as unlabeled data in tranductive learning process.

We investigated two distance measures for LP: cosine similarity and Jensen-Shannon (JS) divergence (Lin, 1991). Cosine similarity measures the angle between two feature vectors, while JS divergence measures the distance between two probability distributions if each feature vector is considered as probability distribution over features.

For sense disambiguation on SENSEVAL-3 data, we constructed connected graphs for LP algorithm following (Niu et al., 2005): two instances u, v will be connected by an edge if u is among v's k nearest neighbors, or if v is among u's k nearest neighbors

[2]Following (Baker and McCallum, 1998), we use LSI as a representative method for unsupervised dimensionality reduction.

[3]Available at http://www.cs.huji.ac.il/∼noamm/

[4]We used SVM^{light} with linear kernel function, available at http://svmlight.joachims.org/.

[5]If there are multiple sense tags for an instance in training set or test set, then only the first tag is considered as correct answer. Furthermore, if the answer of the instance in test set is "U", then this instance will be removed from test set.

as measured by cosine or JS distance measure. k is 5 in later experiments.

4.2 Experiments on Full SENSEVAL-3 Data

In this experiment, we took the training set in SENSEVAL-3 as labeled data, and the test set as unlabeled data. In other words, all of dimensionality reduction methods and classifiers can use the label information in training set, but can not access the label information in test set. We evaluated different sense disambiguation processes using test set in SENSEVAL-3.

We use features with occurrence frequency no less than 3 in training set and test set as feature set F for each ambiguous word. F consists of two disjoint subsets: F_L and $F_{\overline{L}}$. F_L consists of features occurring in training set of target word in SENSEVAL-3, while $F_{\overline{L}}$ consists of features that occur in test set, but never appear in training set.

Table 1 lists accuracies of SVM and LP without or with dimensionality reduction on full SENSEVAL-3 data. From this table, we have some findings as follows:

(1) If without dimensionality reduction, the best performance of sense disambiguation is 70.3% (LP_{JS}), while if using dimensionality reduction, the best two systems can achieve 69.8% ($SuFS + LP_{JS}$) and 69.0% ($SemiFC + LP_{JS}$) accuracies. It seems that feature selection and feature clustering can significantly reduce the dimensionality of feature space while losing only about 1.0% accuracy.

(2) Furthermore, LP_{JS} algorithm performs better than SVM when combined with the same dimensionality reduction technique (except IDC). Notice that LP algorithm uses unlabelled data during its disambiguation phase while SVM doesn't. This indicates that using unlabeled data helps to improve the performance of sense disambiguation.

(3) When using LP algorithm for sense disambiguation, SemiFC performs better than other feature clustering algorithms, such as SuFC, IDC. This indicates that clustering seen and unseen features can satisfy the requirement of semi-supervised learning algorithm, which does help the classification process.

(4) When using SuFC, IDC, SuFS, or SemiFC for dimensionality reduction, the performance of sense disambiguation is always better than that using LSI as dimensionality reduction method. SuFC, IDC, SuFS, and SemiFC use label information to guide feature clustering or feature selection, while LSI is an unsupervised factor analysis method that can conduct dimensionality reduction without the use of label information from labeled data. This indicates that using label information in dimensionality reduction procedure can cluster features into better groups or select better feature subsets, which results in better representation of contexts in reduced feature space.

4.3 Additional Experiments on Sampled SENSEVAL-3 Data

For investigating the performance of various dimensionality reduction techniques with very small training data, we ran them with only l_w examples from training set of each word in SENSEVAL-3 as labeled data. The remaining training examples and all the test examples were used as unlabeled data for SemiFC or LP algorithm. Finally we evaluated different sense disambiguation processes using test set in SENSEVAL-3. For each labeled set size l_w, we performed 20 trials. In each trial, we randomly sampled l_w labeled examples for each word from training set. If any sense was absent from the sampled labeled set, we redid the sampling. l_w is set as $N_{w,train} \times 10\%$, where $N_{w,train}$ is the number of examples in training set of word w. Other settings of this experiment is as same as that of previous one in section 4.2.

In this experiment, feature set F is as same as that in section 4.2. F_L consists of features occurring in sampled training set of target word in SENSEVAL-3, while $F_{\overline{L}}$ consists of features that occur in unlabeled data (including unselected training data and all the test set), but never appear in labeled data (sampled training set).

Table 2 lists accuracies of SVM and LP without or with dimensionality reduction on sampled SENSEVAL-3 training data [6]. From this table, we have some findings as follows:

(1) If without dimensionality reduction, the best performance of sense disambiguation is 54.9% (LP_{JS}), while if using dimensionality reduction, the

[6]We can not obtain the results of IDC over 20 trials since it costs about 50 hours for each trial (Pentium 1.4 GHz CPU/1.0 GB memory).

Table 1: This table lists the accuracies of SVM and LP without or with dimensionality reduction on full SENSEVAL-3 data. There is no result for $LSI + LP_{JS}$, since the vectors obtained by LSI may contain negative values, which prohibits the application of JS divergence for measuring the distance between these vectors.

Classifier	Without dimensionality reduction	With various dimensionality reduction techniques SuFC	IDC	SuFS	LSI	SemiFC
SVM	69.7%	66.4%	65.1%	65.2%	59.1%	64.0%
LP_{cosine}	68.4%	66.7%	64.9%	66.0%	60.7%	67.6%
LP_{JS}	70.3%	67.2%	64.0%	69.8%	-	69.0%

Table 2: This table lists the accuracies of SVM and LP without or with dimensionality reduction on sampled SENSEVAL-3 training data. For each classifier, we performed paired t-test between the system using SemiFC for dimensionality reduction and any other system with or without dimensionality reduction. $\gg$ (or $\ll$) means p-value ≤ 0.01, while $>$ (or $<$) means p-value falling into $(0.01, 0.05]$. Both $\gg$ (or $\ll$) and $>$ (or $<$) indicate that the performance of current WSD system is significantly better (or worse) than that using SemiFC for dimensionality reduction, when given same classifier.

Classifier	Without dimensionality reduction	With various dimensionality reduction techniques SuFC	SuFS	LSI	SemiFC
SVM	53.4±1.1% ($\gg$)	50.4±1.1% ($\ll$)	52.2±1.2% ($>$)	49.8±0.8% ($\ll$)	51.5±1.0%
LP_{cosine}	54.4±1.2% ($\gg$)	49.5±1.1% ($\ll$)	51.1±1.0% ($\ll$)	49.8±1.0% ($\ll$)	52.9±1.0%
LP_{JS}	54.9±1.1% ($\gg$)	52.0±0.9% ($\ll$)	52.5±1.0% ($\ll$)	-	54.1±1.2%

best performance of sense disambiguation is 54.1% ($SemiFC + LP_{JS}$). Feature clustering can significantly reduce the dimensionality of feature space while losing only 0.8% accuracy.

(2) LP_{JS} algorithm performs better than SVM when combined with most of dimensionality reduction techniques. This result confirmed our previous conclusion that using unlabeled data can improve the sense disambiguation process. Furthermore, SemiFC performs significantly better than SuFC and SuFS when using LP as the classifier for sense disambiguation. The reason is that when given very few labeled examples, the distribution of class labels over features can not be reliably estimated, which deteriorates the performance of SuFC or SuFS. But SemiFC uses only raw label information (hard label of each feature) estimated from labeled data, which makes it robust to the noise in very small labeled data.

(3) SuFC, SuFS and SemiFC perform better than LSI no matter which classifier is used for sense disambiguation. This observation confirmed our previous conclusion that using label information to guide dimensionality reduction process can result in better representation of contexts in feature subspace, which further improves the results of sense disambiguation.

5 Related Work

Feature clustering has been extensively studied for the benefit of text categorization and document clustering, which can be categorized as supervised feature clustering, semi-supervised feature clustering, and unsupervised feature clustering.

Supervised feature clustering algorithms (Baker and McCallum, 1998; Bekkerman et al., 2003; Slonim and Tishby, 2001) usually cluster words into groups based on the distribution of class labels over features. Baker and McCallum (1998) apply supervised feature clustering based on distributional clustering for text categorization, which can compress the feature space much more aggressively while still maintaining state of the art classification accuracy. Slonim and Tishby (2001) and Bekkerman et. al. (2003) apply information bottleneck method to find word clusters. They present similar results with the work by Baker and McCallum (1998). Slonim and Tishby (2001) goes further to show that when the training sample is small, word clusters can yield significant improvement in classification accuracy.

Unsupervised feature clustering algorithms (Dhillon, 2001; Dhillon et al., 2002; Dhillon et al., 2003; El-Yaniv and Souroujon, 2001; Slonim and Tishby, 2000) perform word clustering by the use of word-document co-occurrence matrix, which do not utilize class labels to guide clustering process. Slonim and Tishby (2000), El-Yaniv and Souroujon (2001) and Dhillon et. al. (2003) show that word clusters can improve the performance of document clustering.

El-Yaniv and Souroujon (2001) present an iterative double clustering (IDC) algorithm, which performs iterations of double clustering (Slonim and Tishby, 2000). Furthermore, they extend IDC algorithm for semi-supervised learning when given both labeled and unlabeled data.

Our algorithm belongs to the family of semi-supervised feature clustering techniques, which can utilize both labeled and unlabeled data to perform feature clustering.

Supervised feature clustering can not group unseen features (features that do not occur in labeled data) into meaningful clusters since there are no class labels associated with these unseen features. Our algorithm can overcome this problem by inducing class labels for unseen features based on the similarity among seen features and unseen features, then clustering all the features (including both seen features and unseen features) based on the distribution of class labels over them.

Compared with the semi-supervised version of IDC algorithm, our algorithm is more efficient, since we perform feature clustering without iterations.

The difference between our algorithm and unsupervised feature clustering is that our algorithm depends on both labeled and unlabeled data, but unsupervised feature clustering requires only unlabeled data.

O'Hara et. al. (2004) use semantic class-based collocations to augment traditional word-based collocations for supervised WSD. Three separate sources of word relatedness are used for these collocations: 1) WordNet hypernym relations; 2) cluster-based word similarity classes; and 3) dictionary definition analysis. Their system achieved 56.6% fine-grained score on ELS task of SENSEVAL-3. In contrast with their work, our data-driven method for feature clustering based WSD does not require external knowledge resource. Furthermore, our $SemiFC+LP_{JS}$ method can achieve 69.0% fine-grained score on the same dataset, which shows the effectiveness of our method.

6 Conclusion

In this paper we have investigated feature clustering techniques for WSD, which usually group features into clusters based on the distribution of class labels over features. We propose a semi-supervised feature clustering algorithm to satisfy the requirement of semi-supervised classification algorithms for dimensionality reduction in feature space. Our experimental results on SENSEVAL-3 data show that feature clustering can aggressively reduce the dimensionality of feature space while still maintaining state of the art sense disambiguation accuracy. Furthermore, when combined with a semi-supervised WSD algorithm, semi-supervised feature clustering outperforms supervised feature clustering and other dimensionality reduction techniques. Our additional experiments on sampled SENSEVAL-3 data indicate that our semi-supervised feature clustering method is robust to the noise in small labeled data, which achieves better performance than supervised feature clustering.

In the future, we may extend our work by using more datasets to empirically evaluate this feature clustering algorithm. This semi-supervised feature clustering framework is quite general, which can be applied to other NLP tasks, ex. text categorization.

Acknowledgements We would like to thank anonymous reviewers for their helpful comments. Z.Y. Niu is supported by A*STAR Graduate Scholarship.

References

Baker L. & McCallum A.. 1998. Distributional Clustering of Words for Text Classification. *ACM SIGIR*

1998.

Bekkerman, R., El-Yaniv, R., Tishby, N., & Winter, Y.. 2003. Distributional Word Clusters vs. Words for Text Categorization. *Journal of Machine Learning Research*, Vol. 3: 1183-1208.

Dagan, I. & Itai A.. 1994. Word Sense Disambiguation Using A Second Language Monolingual Corpus. *Computational Linguistics*, Vol. 20(4), pp. 563-596.

Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., & Harshman, R.A.. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, Vol. 41(6), pp. 391-407.

Dhillon I.. 2001. Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning. *ACM SIGKDD 2001*.

Dhillon I., Mallela S., & Kumar R.. 2002. Enhanced Word Clustering for Hierarchical Text Classification. *ACM SIGKDD 2002*.

Dhillon I., Mallela S., & Modha, D.. 2003. Information-Theoretic Co-Clustering. *ACM SIGKDD 2003*.

El-Yaniv, R., & Souroujon, O.. 2001. Iterative Double Clustering for Unsupervised and Semi-Supervised Learning. *NIPS 2001*.

Forman, G.. 2003. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research* 3(Mar):1289–1305.

Leacock, C., Miller, G.A. & Chodorow, M.. 1998. Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 24:1, 147–165.

Lee, Y.K. & Ng, H.T.. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. *EMNLP 2002*, (pp. 41-48).

Lesk M.. 1986. Automated Word Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. *ACM SIGDOC 1986*.

Li, H. & Li, C.. 2004. Word Translation Disambiguation Using Bilingual Bootstrapping. *Computational Linguistics*, 30(1), 1-22.

Lin, J. 1991. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*, 37:1, 145–150.

McCarthy, D., Koeling, R., Weeds, J., & Carroll, J.. 2004. Finding Predominant Word Senses in Untagged Text. *ACL 2004*.

Mihalcea R.. 2002. Instance Based Learning with Automatic Feature Selection Applied to Word Sense Disambiguation. *COLING 2002*.

Mihalcea R.. 2004. Co-Training and Self-Training for Word Sense Disambiguation. *CoNLL 2004*.

Mihalcea R., Chklovski, T., & Kilgariff, A.. 2004. The SENSEVAL-3 English Lexical Sample Task. *SENSEVAL 2004*.

Niu, Z.Y., Ji, D.H., & Tan, C.L.. 2004. Learning Word Senses With Feature Selection and Order Identification Capabilities. *ACL 2004*.

Niu, Z.Y., Ji, D.H., & Tan, C.L.. 2005. Word Sense Disambiguation Using Label Propagation Based Semi-Supervised Learning. *ACL 2005*.

O'Hara, T., Bruce, R., Donner, J., & Wiebe, J.. 2004. Class-Based Collocations for Word-Sense Disambiguation. *SENSEVAL 2004*.

Park, S.B., Zhang, B.T., & Kim, Y.T.. 2000. Word Sense Disambiguation by Learning from Unlabeled Data. *ACL 2000*.

Pedersen. T., & Bruce, R.. 1997. Distinguishing Word Senses in Untagged Text. *EMNLP 1997*.

Sch$\ddot{u}$tze, H.. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24:1, 97–123.

Slonim, N. & Tishby, N.. 2000. Document Clustering Using Word Clusters via the Information Bottleneck Method. *ACM SIGIR 2000*.

Slonim, N. & Tishby, N.. 2001. The Power of Word Clusters for Text Classification. *The 23rd European Colloquium on Information Retrieval Research*.

Towel, G. & Voorheest, E.M.. 1998. Disambiguating Highly Ambiguous Words. *Computational Linguistics*, 24:1, 125–145.

Yarowsky, D.. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *ACL 1995*, pp. 189-196.

Zhou D., Bousquet, O., Lal, T.N., Weston, J., & Sch$\ddot{o}$lkopf, B.. 2003. Learning with Local and Global Consistency. *NIPS 16*,pp. 321-328.

Zhu, X. & Ghahramani, Z.. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. *CMU CALD tech report CMU-CALD-02-107*.

Zhu, X., Ghahramani, Z., & Lafferty, J.. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. *ICML 2003*.

Using Random Walks for Question-focused Sentence Retrieval

Jahna Otterbacher[1], **Güneş Erkan**[2], **Dragomir R. Radev**[1,2]
[1]School of Information, [2]Department of EECS
University of Michigan
{jahna,gerkan,radev}@umich.edu

Abstract

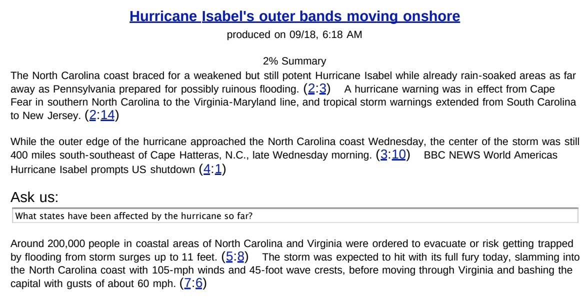
We consider the problem of question-focused sentence retrieval from complex news articles describing multi-event stories published over time. Annotators generated a list of questions central to understanding each story in our corpus. Because of the dynamic nature of the stories, many questions are time-sensitive (e.g. "How many victims have been found?") Judges found sentences providing an answer to each question. To address the sentence retrieval problem, we apply a stochastic, graph-based method for comparing the relative importance of the textual units, which was previously used successfully for generic summarization. Currently, we present a topic-sensitive version of our method and hypothesize that it can outperform a competitive baseline, which compares the similarity of each sentence to the input question via IDF-weighted word overlap. In our experiments, the method achieves a TRDR score that is significantly higher than that of the baseline.

1 Introduction

Recent work has motivated the need for systems that support "Information Synthesis" tasks, in which a user seeks a global understanding of a topic or story (Amigo et al., 2004). In contrast to the classical question answering setting (e.g. TREC-style Q&A (Voorhees and Tice, 2000)), in which the user presents a single question and the system returns a corresponding answer (or a set of likely answers), in this case the user has a more complex information need.

Similarly, when reading about a complex news story, such as an emergency situation, users might seek answers to a set of questions in order to understand it better. For example, Figure 1 shows the interface to our Web-based news summarization system, which a user has queried for information about Hurricane Isabel. Understanding such stories is challenging for a number of reasons. In particular, complex stories contain many sub-events (e.g. the devastation of the hurricane, the relief effort, etc.) In addition, while some facts surrounding the situation do not change (such as "Which area did the hurricane first hit?"), others may change with time ("How many people have been left homeless?"). Therefore, we are working towards developing a system for question answering from clusters of complex stories published over time. As can be seen at the bottom of Figure 1, we plan to add a component to our current system that allows users to ask questions as they read a story. They may then choose to receive either a precise answer or a question-focused summary.

Currently, we address the question-focused sentence retrieval task. While passage retrieval (PR) is clearly not a new problem (e.g. (Robertson et al., 1992; Salton et al., 1993)), it remains important and yet often overlooked. As noted by (Gaizauskas et al., 2004), while PR is the crucial first step for question answering, Q&A research has typically not empha-



Hurricane Isabel's outer bands moving onshore

produced on 09/18, 6:18 AM

2% Summary

The North Carolina coast braced for a weakened but still potent Hurricane Isabel while already rain-soaked areas as far away as Pennsylvania prepared for possibly ruinous flooding. (2:3) A hurricane warning was in effect from Cape Fear in southern North Carolina to the Virginia-Maryland line, and tropical storm warnings extended from South Carolina to New Jersey. (2:14)

While the outer edge of the hurricane approached the North Carolina coast Wednesday, the center of the storm was still 400 miles south-southeast of Cape Hatteras, N.C., late Wednesday morning. (3:10) BBC NEWS World Americas Hurricane Isabel prompts US shutdown (4:1)

Ask us:

What states have been affected by the hurricane so far?

Around 200,000 people in coastal areas of North Carolina and Virginia were ordered to evacuate or risk getting trapped by flooding from storm surges up to 11 feet. (5:8) The storm was expected to hit with its full fury today, slamming into the North Carolina coast with 105-mph winds and 45-foot wave crests, before moving through Virginia and bashing the capital with gusts of about 60 mph. (7:6)

Figure 1: Question tracking interface to a summarization system.

sized it. The specific problem we consider differs from the classic task of PR for a Q&A system in interesting ways, due to the time-sensitive nature of the stories in our corpus. For example, one challenge is that the answer to a user's question may be updated and reworded over time by journalists in order to keep a running story fresh, or because the facts themselves change. Therefore, there is often more than one correct answer to a question.

We aim to develop a method for sentence retrieval that goes beyond finding sentences that are similar to a single query. To this end, we propose to use a stochastic, graph-based method. Recently, graph-based methods have proved useful for a number of NLP and IR tasks such as document re-ranking in ad hoc IR (Kurland and Lee, 2005) and analyzing sentiments in text (Pang and Lee, 2004). In (Erkan and Radev, 2004), we introduced the LexRank method and successfully applied it to generic, multi-document summarization. Presently, we introduce topic-sensitive LexRank in creating a sentence retrieval system. We evaluate its performance against a competitive baseline, which considers the similarity between each sentence and the question (using IDF-weighed word overlap). We demonstrate that LexRank significantly improves question-focused sentence selection over the baseline.

2 Formal description of the problem

Our goal is to build a question-focused sentence retrieval mechanism using a topic-sensitive version of the LexRank method. In contrast to previous PR systems such as Okapi (Robertson et al., 1992), which ranks documents for relevancy and then proceeds to find paragraphs related to a question, we address the finer-grained problem of finding sentences containing answers. In addition, the input to our system is a set of documents relevant to the topic of the query that the user has already identified (e.g. via a search engine). Our system does not rank the input documents, nor is it restricted in terms of the number of sentences that may be selected from the same document.

The output of our system, a ranked list of sentences relevant to the user's question, can be subsequently used as input to an answer selection system in order to find specific answers from the extracted sentences. Alternatively, the sentences can be returned to the user as a question-focused summary. This is similar to "snippet retrieval" (Wu et al., 2004). However, in our system answers are extracted from a set of multiple documents rather than on a document-by-document basis.

3 Our approach: topic-sensitive LexRank

3.1 The LexRank method

In (Erkan and Radev, 2004), the concept of graph-based centrality was used to rank a set of sentences, in producing generic multi-document summaries. To apply LexRank, a similarity graph is produced for the sentences in an input document set. In the graph, each node represents a sentence. There are edges between nodes for which the cosine similarity between the respective pair of sentences exceeds a given threshold. The degree of a given node is an indication of how much information the respective sentence has in common with other sentences. Therefore, sentences that contain the most salient information in the document set should be very central within the graph.

Figure 2 shows an example of a similarity graph for a set of five input sentences, using a cosine similarity threshold of 0.15. Once the similarity graph is constructed, the sentences are then ranked according to their eigenvector centrality. As previously mentioned, the original LexRank method performed well in the context of generic summarization. Below, we describe a topic-sensitive version of LexRank, which is more appropriate for the question-focused sentence retrieval problem. In the new approach, the score of a sentence is determined by a mixture model of the relevance of the sentence to the query and the similarity of the sentence to other high-scoring sentences.

3.2 Relevance to the question

In topic-sensitive LexRank, we first stem all of the sentences in a set of articles and compute word IDFs by the following formula:

$$\text{idf}_w = \log\left(\frac{N+1}{0.5 + sf_w}\right) \tag{1}$$

where N is the total number of sentences in the cluster, and sf_w is the number of sentences that the word w appears in.

We also stem the question and remove the stop words from it. Then the relevance of a sentence s to the question q is computed by:

$$\text{rel}(s|q) = \sum_{w \in q} \log(tf_{w,s} + 1) \times \log(tf_{w,q} + 1) \times \text{idf}_w \tag{2}$$

where $tf_{w,s}$ and $tf_{w,q}$ are the number of times w appears in s and q, respectively. This model has proven to be successful in query-based sentence retrieval (Allan et al., 2003), and is used as our competitive baseline in this study (e.g. Tables 4, 5 and 7).

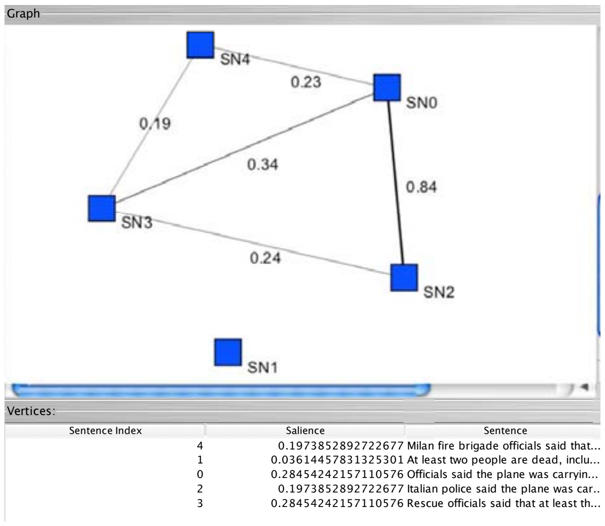
3.3 The mixture model

The baseline system explained above does not make use of any inter-sentence information in a cluster. We hypothesize that a sentence that is similar to the high scoring sentences in the cluster should also have a high score. For instance, if a sentence that gets a high score in our baseline model is likely to contain an answer to the question, then a related sentence, which may not be similar to the question itself, is also likely to contain an answer.

This idea is captured by the following mixture model, where $p(s|q)$, the score of a sentence s given a question q, is determined as the sum of its relevance to the question (using the same measure as the baseline described above) and the similarity to the other sentences in the document cluster:

$$p(s|q) = d\frac{\text{rel}(s|q)}{\sum_{z \in C}\text{rel}(z|q)} + (1-d)\sum_{v \in C}\frac{\text{sim}(s,v)}{\sum_{z \in C} sim(z,v)}p(v|q) \tag{3}$$

where C is the set of all sentences in the cluster. The value of d, which we will also refer to as the "question bias," is a trade-off between two terms in the



Vertices:

Sentence Index	Salience	Sentence
4	0.1973852892722677	Milan fire brigade officials said that...
1	0.03614457831325301	At least two people are dead, inclu...
0	0.28454242157110576	Officials said the plane was carryin...
2	0.1973852892722677	Italian police said the plane was car..
3	0.28454242157110576	Rescue officials said that at least th...

Figure 2: LexRank example: sentence similarity graph with a cosine threshold of 0.15.

equation and is determined empirically. For higher values of d, we give more importance to the relevance to the question compared to the similarity to the other sentences in the cluster. The denominators in both terms are for normalization, which are described below. We use the cosine measure weighted by word IDFs as the similarity between two sentences in a cluster:

$$\text{sim}(x,y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}} \tag{4}$$

Equation 3 can be written in matrix notation as follows:

$$\mathbf{p} = [d\mathbf{A} + (1-d)\mathbf{B}]^{\text{T}}\mathbf{p} \tag{5}$$

$\mathbf{A}$ is the square matrix such that for a given index i, all the elements in the i^{th} column are proportional to $\text{rel}(i|q)$. $\mathbf{B}$ is also a square matrix such that each entry $\mathbf{B}(i,j)$ is proportional to $sim(i,j)$. Both matrices are normalized so that row sums add up to 1. Note that as a result of this normalization, all rows of the resulting square matrix $\mathbf{Q} = [d\mathbf{A} + (1-d)\mathbf{B}]$ also add up to 1. Such a matrix is called *stochastic* and defines a Markov chain. If we view each sentence as a state in a Markov chain, then $\mathbf{Q}(i,j)$ specifies the transition probability from state i to state j in the corresponding Markov chain. The vector $\mathbf{p}$ we are looking for in Equation 5 is the stationary distribution of the Markov chain. An intuitive interpretation of the stationary distribution can be understood by the concept of a random walk on the graph representation of the Markov chain.

With probability d, a transition is made from the current node (sentence) to the nodes that are similar to the query. With probability (1-d), a transition is made to the nodes that are lexically similar to the current node. Every transition is weighted according to the similarity distributions. Each element of the vector $\mathbf{p}$ gives the asymptotic probability of ending up at the corresponding state in the long run regardless of the starting state. The stationary distribution of a Markov chain can be computed by a simple iterative algorithm, called power method.[1]

A simpler version of Equation 5, where $\mathbf{A}$ is a uniform matrix and $\mathbf{B}$ is a normalized binary matrix, is known as PageRank (Brin and Page, 1998; Page et al., 1998) and used to rank the web pages by the Google search engine. It was also the model used to rank sentences in (Erkan and Radev, 2004).

3.4 Experiments with topic-sensitive LexRank

We experimented with different values of d on our training data. We also considered several threshold values for inter-sentence cosine similarities, where we ignored the similarities between the sentences that are below the threshold. In the training phase of the experiment, we evaluated all combinations of LexRank with d in the range of $[0, 1]$ (in increments of 0.10) and with a similarity threshold ranging from $[0, 0.9]$ (in increments of 0.05). We then found all configurations that outperformed the baseline. These configurations were then applied to our development/test set. Finally, our best sentence retrieval system was applied to our test data set and evaluated against the baseline. The remainder of the paper will explain this process and the results in detail.

4 Experimental setup

4.1 Corpus

We built a corpus of 20 multi-document clusters of complex news stories, such as plane crashes, political controversies and natural disasters. The data clusters and their characteristics are shown in Table 1. The news articles were collected from various sources. "Newstracker" clusters were collected automatically by our Web-based news summarization system. The number of clusters randomly assigned to the training, development/test and test data sets were 11, 3 and 6, respectively.

Next, we assigned each cluster of articles to an annotator, who was asked to read all articles in the cluster. He or she then generated a list of factual questions key to understanding the story. Once we collected the questions for each cluster, two judges independently annotated nine of the training clusters. For each sentence and question pair in a given cluster, the judges were asked to indicate whether or not the sentence contained a complete answer to the question. Once an acceptable rate of interjudge agreement was verified on the first nine clusters (Kappa (Carletta, 1996) of 0.68), the remaining 11 clusters were annotated by one judge each.

In some cases, the judges did not find any sentences containing the answer for a given question. Such questions were removed from the corpus. The final number of questions annotated for answers over the entire corpus was 341, and the distributions of questions per cluster can be found in Table 1.

4.2 Evaluation metrics and methods

To evaluate our sentence retrieval mechanism, we produced extract files, which contain a list of sentences deemed to be relevant to the question, for the system and from human judgment. To compare different configurations of our system to the baseline system, we produced extracts at a fixed length of 20 sentences. While evaluations of question answering systems are often based on a shorter list of ranked sentences, we chose to generate longer lists for several reasons. One is that we are developing a PR system, of which the output can then be input to an answer extraction system for further processing. In such a setting, we would most likely want to generate a relatively longer list of candidate sentences. As previously mentioned, in our corpus the questions often have more than one relevant answer, so ideally, our PR system would find many of the relevant sentences, sending them on to the answer component to decide which answer(s) should be returned to the user. Each system's extract file lists the document

[1]The stationary distribution is unique and the power method is guaranteed to converge provided that the Markov chain is ergodic (Seneta, 1981). A non-ergodic Markov chain can be made ergodic by reserving a small probability for jumping to any other state from the current state (Page et al., 1998).

Cluster	Sources	Articles	Questions	Data set	Sample question
Algerian terror threat	AFP, UPI	2	12	train	What is the condition under which GIA will take its action?
Milan plane crash	MSNBC, CNN, ABC, Fox, USAToday	9	15	train	How many people were in the building at the time of the crash?
Turkish plane crash	BBC, ABC, FoxNews, Yahoo	10	12	train	To where was the plane headed?
Moscow terror attack	UPI, AFP, AP	7	7	train	How many people were killed in the most recent explosion?
Rhode Island club fire	MSNBC, CNN, ABC, Lycos, Fox, BBC, Ananova	10	8	train	Who was to blame for the fire?
FBI most wanted	AFP, UPI	3	14	train	How much is the State Department offering for information leading to bin Laden's arrest?
Russia bombing	AP, AFP	2	11	train	What was the cause of the blast?
Bali terror attack	CNN, FoxNews, ABC, BBC, Ananova	10	30	train	What were the motivations of the attackers?
Washington DC sniper	FoxNews, Ha'aretz, BBC, BBC, Washington Times, CBS	8	28	train	What kinds of equipment or weapons were used in the killings?
GSPC terror group	Newstracker	8	29	train	What are the charges against the GSPC suspects?
China earthquake	Novelty 43	25	18	train	What was the magnitude of the earthquake in Zhangjiakou?
Gulfair	ABC, BBC, CNN, USAToday, FoxNews, Washington Post	11	29	dev/test	How many people were on board?
David Beckham trade	AFP	20	28	dev/test	How long had Beckham been playing for MU before he moved to RM?
Miami airport evacuation	Newstracker	12	15	dev/test	How many concourses does the airport have?
US hurricane	DUC d04a	14	14	test	In which places had the hurricane landed?
EgyptAir crash	Novelty 4	25	29	test	How many people were killed?
Kursk submarine	Novelty 33	25	30	test	When did the Kursk sink?
Hebrew University bombing	Newstracker	11	27	test	How many people were injured?
Finland mall bombing	Newstracker	9	15	test	How many people were in the mall at the time of the bombing?
Putin visits England	Newstracker	12	20	test	What issue concerned British human rights groups?

Table 1: Corpus of complex news stories.

and sentence numbers of the top 20 sentences. The "gold standard" extracts list the sentences judged as containing answers to a given question by the annotators (and therefore have variable sizes) in no particular order.[2]

We evaluated the performance of the systems using two metrics - Mean Reciprocal Rank (MRR) (Voorhees and Tice, 2000) and Total Reciprocal Document Rank (TRDR) (Radev et al., 2005). MRR, used in the TREC Q&A evaluations, is the reciprocal rank of the first correct answer (or sentence, in our case) to a given question. This measure gives us an idea of how far down we must look in the ranked list in order to find a correct answer. To contrast, TRDR is the total of the reciprocal ranks of all answers found by the system. In the context of answering questions from complex stories, where there is often more than one correct answer to a question, and where answers are typically time-dependent, we should focus on maximizing TRDR, which gives us a measure of how many of the relevant sentences were identified by the system. However, we report both the average MRR and TRDR over all questions in a given data set.

5 LexRank versus the baseline system

In the training phase, we searched the parameter space for the values of d (the question bias) and the similarity threshold in order to optimize the resulting TRDR scores. For our problem, we expected that a relatively low similarity threshold pair with a high question bias would achieve the best results. Table 2 shows the effect of varying the similarity threshold.[3] The notation $LR[a, d]$ is used, where a is the similarity threshold and d is the question bias. The optimal range for the parameter a was between 0.14 and 0.20. This is intuitive because if the threshold is too high, such that only the most lexically similar sentences are represented in the graph, the method does not find sentences that are related but are more lex-

[2]For clusters annotated by two judges, all sentences chosen by at least one judge were included.

[3]A threshold of -1 means that no threshold was used such that all sentences were included in the graph.

System	Ave. MRR	Ave. TRDR
LR[-1.0,0.65]	0.5270	0.8117
LR[0.02,0.65]	0.5261	0.7950
LR[0.16,0.65]	0.5131	0.8134
LR[0.18,0.65]	0.5062	0.8020
LR[0.20,0.65]	0.5091	0.7944
LR[-1.0,0.80]	0.5288	0.8152
LR[0.02,0.80]	0.5324	0.8043
LR[0.16,0.80]	0.5184	0.8160
LR[0.18,0.80]	0.5199	0.8154
LR[0.20,0.80]	0.5282	0.8152

Table 2: Training phase: effect of similarity threshold (a) on Ave. MRR and TRDR.

System	Ave. MRR	Ave. TRDR
LR[0.02,0.65]	0.5261	0.7950
LR[0.02,0.70]	0.5290	0.7997
LR[0.02,0.75]	0.5299	0.8013
LR[0.02,0.80]	0.5324	0.8043
LR[0.02,0.85]	0.5322	0.8038
LR[0.02,0.90]	0.5323	0.8077
LR[0.20,0.65]	0.5091	0.7944
LR[0.20,0.70]	0.5244	0.8105
LR[0.20,0.75]	0.5285	0.8137
LR[0.20,0.80]	0.5282	0.8152
LR[0.20,0.85]	0.5317	0.8203
LR[0.20,0.90]	0.5368	0.8265

Table 3: Training phase: effect of question bias (d) on Ave. MRR and TRDR.

ically diverse (e.g. paraphrases). Table 3 shows the effect of varying the question bias at two different similarity thresholds (0.02 and 0.20). It is clear that a high question bias is needed. However, a small probability for jumping to a node that is lexically similar to the given sentence (rather than the question itself) is needed. Table 4 shows the configurations of LexRank that performed better than the baseline system on the training data, based on mean TRDR scores over the 184 training questions. We applied all four of these configurations to our unseen development/test data, in order to see if we could further differentiate their performances.

5.1 Development/testing phase

The scores for the four LexRank systems and the baseline on the development/test data are shown in

System	Ave. MRR	Ave. TRDR
Baseline	0.5518	0.8297
LR[0.14,0.95]	0.5267	0.8305
LR[0.18,0.90]	0.5376	0.8382
LR[0.18,0.95]	0.5421	0.8382
LR[0.20,0.95]	0.5404	0.8311

Table 4: Training phase: systems outperforming the baseline in terms of TRDR score.

System	Ave. MRR	Ave. TRDR
Baseline	0.5709	1.0002
LR[0.14,0.95]	0.5882	1.0469
LR[0.18,0.90]	0.5820	1.0288
LR[0.18,0.95]	0.5956	1.0411
LR[0.20,0.95]	0.6068	1.0601

Table 5: Development testing evaluation.

Cluster	B-MRR	LR-MRR	B-TRDR	LR-TRDR
Gulfair	0.5446	0.5461	0.9116	0.9797
David Beckham trade	0.5074	0.5919	0.7088	0.7991
Miami airport evacuation	0.7401	0.7517	1.7157	1.7028

Table 6: Average scores by cluster: baseline versus LR[0.20,0.95].

Table 5. This time, all four LexRank systems outperformed the baseline, both in terms of average MRR and TRDR scores. An analysis of the average scores over the 72 questions within each of the three clusters for the best system, LR[0.20,0.95], is shown in Table 6. While LexRank outperforms the baseline system on the first two clusters both in terms of MRR and TRDR, their performances are not substantially different on the third cluster. Therefore, we examined properties of the questions within each cluster in order to see what effect they might have on system performance.

We hypothesized that the baseline system, which compares the similarity of each sentence to the question using IDF-weighted word overlap, should perform well on questions that provide many content words. To contrast, LexRank might perform better when the question provides fewer content words, since it considers both similarity to the query and inter-sentence similarity. Out of the 72 questions in the development/test set, the baseline system outperformed LexRank on 22 of the questions. In fact, the average number of content words among these 22 questions was slightly, but not significantly, higher than the average on the remaining questions (3.63 words per question versus 3.46). Given this observation, we experimented with two mixed strategies, in which the number of content words in a question determined whether LexRank or the baseline system was used for sentence retrieval. We tried threshold values of 4 and 6 content words, however, this did not improve the performance over the pure strategy of system LR[0.20,0.95]. Therefore, we applied this

	Ave. MRR	**Ave. TRDR**
Baseline	0.5780	0.8673
LR[0.20,0.95]	0.6189	0.9906
p-value	na	0.0619

Table 7: Testing phase: baseline vs. LR[0.20,0.95].

system versus the baseline to our unseen test set of 134 questions.

5.2 Testing phase

As shown in Table 7, LR[0.20,0.95] outperformed the baseline system on the test data both in terms of average MRR and TRDR scores. The improvement in average TRDR score was statistically significant with a p-value of 0.0619. Since we are interested in a passage retrieval mechanism that finds sentences relevant to a given question, providing input to the question answering component of our system, the improvement in average TRDR score is very promising. While we saw in Section 5.1 that LR[0.20,0.95] may perform better on some question or cluster types than others, we conclude that it beats the competitive baseline when one is looking to optimize mean TRDR scores over a large set of questions. However, in future work, we will continue to improve the performance, perhaps by developing mixed strategies using different configurations of LexRank.

6 Discussion

The idea behind using LexRank for sentence retrieval is that a system that considers only the similarity between candidate sentences and the input query, and not the similarity between the candidate sentences themselves, is likely to miss some important sentences. When using any metric to compare sentences and a query, there is always likely to be a tie between multiple sentences (or, similarly, there may be cases where fewer than the number of desired sentences have similarity scores above zero). LexRank effectively provides a means to break such ties. An example of such a scenario is illustrated in Tables 8 and 9, which show the top ranked sentences by the baseline and LexRank, respectively for the question “What caused the Kursk to sink?” from the Kursk submarine cluster. It can be seen that all top five sentences chosen by the baseline system have

Rank	**Sentence**	**Score**	**Relevant?**
1	The Russian governmental commission on the accident of the submarine Kursk sinking in the Barents Sea on August 12 has rejected 11 original explanations for the disaster, but still cannot conclude what caused the tragedy indeed, Russian Deputy Premier Ilya Klebanov said here Friday.	4.2282	N
2	There has been no final word on what caused the submarine to sink while participating in a major naval exercise, but Defense Minister Igor Sergeyev said the theory that Kursk may have collided with another object is receiving increasingly concrete confirmation.	4.2282	N
3	Russian Deputy Prime Minister Ilya Klebanov said Thursday that collision with a big object caused the Kursk nuclear submarine to sink to the bottom of the Barents Sea.	4.2282	Y
4	Russian Deputy Prime Minister Ilya Klebanov said Thursday that collision with a big object caused the Kursk nuclear submarine to sink to the bottom of the Barents Sea.	4.2282	Y
5	President Clinton’s national security adviser, Samuel Berger, has provided his Russian counterpart with a written summary of what U.S. naval and intelligence officials believe caused the nuclear-powered submarine Kursk to sink last month in the Barents Sea, officials said Wednesday.	4.2282	N

Table 8: Top ranked sentences using baseline system on the question “What caused the Kursk to sink?”.

the same sentence score (similarity to the query), yet the top ranking two sentences are not actually relevant according to the judges. To contrast, LexRank achieved a better ranking of the sentences since it is better able to differentiate between them. It should be noted that both for the LexRank and baseline systems, chronological ordering of the documents and sentences is preserved, such that in cases where two sentences have the same score, the one published earlier is ranked higher.

7 Conclusion

We presented topic-sensitive LexRank and applied it to the problem of sentence retrieval. In a Web-based news summarization setting, users of our system could choose to see the retrieved sentences (as in Table 9) as a question-focused summary. As indicated in Table 9, each of the top three sentences were judged by our annotators as providing a complete answer to the respective question. While the first two sentences provide the same answer (a collision caused the Kursk to sink), the third sentence provides a different answer (an explosion caused the disaster). While the last two sentences do not provide answers according to our judges, they do provide context information about the situation. Alternatively, the user might prefer to see the extracted

Rank	Sentence	Score	Relevant?
1	Russian Deputy Prime Minister Ilya Klebanov said Thursday that collision with a big object caused the Kursk nuclear submarine to sink to the bottom of the Barents Sea.	0.0133	Y
2	Russian Deputy Prime Minister Ilya Klebanov said Thursday that collision with a big object caused the Kursk nuclear submarine to sink to the bottom of the Barents Sea.	0.0133	Y
3	The Russian navy refused to confirm this, but officers have said an explosion in the torpedo compartment at the front of the submarine apparently caused the Kursk to sink.	0.0125	Y
4	President Clinton's national security adviser, Samuel Berger, has provided his Russian counterpart with a written summary of what U.S. naval and intelligence officials believe caused the nuclear-powered submarine Kursk to sink last month in the Barents Sea, officials said Wednesday.	0.0124	N
5	There has been no final word on what caused the submarine to sink while participating in a major naval exercise, but Defense Minister Igor Sergeyev said the theory that Kursk may have collided with another object is receiving increasingly concrete confirmation.	0.0123	N

Table 9: Top ranked sentences using the LR[0.20,0.95] system on the question "What caused the Kursk to sink?"

answers from the retrieved sentences. In this case, the sentences selected by our system would be sent to an answer identification component for further processing. As discussed in Section 2, our goal was to develop a topic-sensitive version of LexRank and to use it to improve a baseline system, which had previously been used successfully for query-based sentence retrieval (Allan et al., 2003). In terms of this task, we have shown that over a large set of unaltered questions written by our annotators, LexRank can, on average, outperform the baseline system, particularly in terms of TRDR scores.

8 Acknowledgments

We would like to thank the members of the CLAIR group at Michigan and in particular Siwei Shen and Yang Ye for their assistance with this project.

References

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 314–321. ACM Press.

Enrique Amigo, Julio Gonzalo, Victor Peinado, Anselmo Peñas, and Felisa Verdejo. 2004. An Empirical Study of Information Synthesis Task. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 207–214, Barcelona, Spain, July.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Jean Carletta. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *CL*, 22(2):249–254.

Gunes Erkan and Dragomir Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text. *JAIR*, 22:457–479.

Robert Gaizauskas, Mark Hepple, and Mark Greenwood. 2004. Information Retrieval for Question Answering: a SIGIR 2004 Workshop. In *SIGIR 2004 Workshop on Information Retrieval for Question Answering*.

Oren Kurland and Lillian Lee. 2005. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *SIGIR 2005*, Salvador, Brazil, August.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford University, Stanford, CA.*

Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Association for Computational Linguistics.*

Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2005. Probabilistic Question Answering on the Web. *Journal of the American Society for Information Science and Technology*, 56(3), March.

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. 1992. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30.

G. Salton, J. Allan, and C. Buckley. 1993. Approaches to Passage REtrieval in Full Text Information Systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58.

E. Seneta. 1981. *Non-negative matrices and markov chains.* Springer-Verlag, New York.

Ellen Voorhees and Dawn Tice. 2000. The TREC-8 Question Answering Track Evaluation. In *Text Retrieval Conference TREC-8*, Gaithersburg, MD.

Harris Wu, Dragomir R. Radev, and Weiguo Fan. 2004. Towards Answer-focused Summarization Using Search Engines. *New Directions in Question Answering.*

Multi-Perspective Question Answering Using the OpQA Corpus

Veselin Stoyanov and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY 14850, USA
{ves,cardie}@cs.cornell.edu

Janyce Wiebe
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
wiebe@cs.pitt.edu

Abstract

We investigate techniques to support the answering of opinion-based questions. We first present the OpQA corpus of opinion questions and answers. Using the corpus, we compare and contrast the properties of fact and opinion questions and answers. Based on the disparate characteristics of opinion vs. fact answers, we argue that traditional fact-based QA approaches may have difficulty in an MPQA setting without modification. As an initial step towards the development of MPQA systems, we investigate the use of machine learning and rule-based subjectivity and opinion source filters and show that they can be used to guide MPQA systems.

1 Introduction

Much progress has been made in recent years in automatic, open-domain question answering (e.g., Voorhees (2001), Voorhees (2002), Voorhees and Buckland (2003)). The bulk of the research in this area, however, addresses fact-based questions like: "When did McDonald's open its first restaurant?" or "What is the Kyoto Protocol?". To date, however, relatively little research been done in the area of Multi-Perspective Question Answering (MPQA), which targets questions of the following sort:

- How is Bush's decision not to ratify the Kyoto Protocol looked upon by Japan and other US allies?
- How do the Chinese regard the human rights record of the United States?

In comparison to fact-based question answering (QA), researchers understand far less about the properties of questions and answers in MPQA, and have yet to develop techniques to exploit knowledge of those properties. As a result, it is unclear whether approaches that have been successful in the domain of fact-based QA will work well for MPQA.

We first present the *OpQA* corpus of opinion questions and answers. Using the corpus, we compare and contrast the properties of fact and opinion questions and answers. We find that text spans identified as answers to opinion questions: (1) are approximately twice as long as those of fact questions, (2) are much more likely (37% vs. 9%) to represent *partial* answers rather than complete answers, (3) vary much more widely with respect to syntactic category – covering clauses, verb phrases, prepositional phrases, and noun phrases; in contrast, fact answers are overwhelming associated with noun phrases, and (4) are roughly half as likely to correspond to a single syntactic constituent type (16-38% vs. 31-53%).

Based on the disparate characteristics of opinion vs. fact answers, we argue that traditional fact-based QA approaches may have difficulty in an MPQA setting without modification. As one such modification, we propose that MPQA systems should rely on natural language processing methods to identify information about opinions. In experiments in opinion question answering using the OpQA corpus, we find that filtering potential answers using machine learning and rule-based NLP opinion filters substantially improves the performance of an end-to-end MPQA system according to both a mean reciprocal rank (MRR) measure (0.59 vs. a baseline of 0.42) and a metric that determines the mean rank of the first correct answer (MRFA) (26.2 vs. a baseline of 61.3). Further, we find that requiring opinion answers to match the requested opinion source (e.g., does <source> approve of the Kyoto Protocol) dramatically improves the performance of the MPQA system on the hardest questions in the corpus.

The remainder of the paper is organized as follows. In the next section we summarize related work. Section 3 describes the OpQA corpus. Section 4 uses the OpQA corpus to identify potentially problematic issues for handling opinion vs. fact questions. Section 5 briefly describes an opinion annotation scheme used in the experiments. Sections 6 and 7 explore the use of opinion information in the design of MPQA systems.

2 Related Work

There is a growing interest in methods for the automatic identification and extraction of opinions, emotions, and sentiments in text. Much of the relevant research explores sentiment classification, a text categorization task in which the goal is to assign to a document either positive ("thumbs up") or negative ("thumbs down") polarity (e.g. Das and Chen (2001), Pang et al. (2002), Turney (2002), Dave et al. (2003), Pang and Lee (2004)). Other research has concentrated on analyzing opinions at, or below, the sentence level. Recent work, for example, indicates that systems can be trained to recognize opinions, their polarity, their source, and their strength to a reasonable degree of accuracy (e.g. Dave et al. (2003), Riloff and Wiebe (2003), Bethard et al. (2004), Pang and Lee (2004), Wilson et al. (2004), Yu and Hatzivassiloglou (2003), Wiebe and Riloff (2005)).

Related work in the area of corpus development includes Wiebe et al.'s (2005) opinion annotation scheme to identify *subjective expressions* — expressions used to express opinions, emotions, sentiments and other *private states* in text. Wiebe et al. have applied the annotation scheme to create the MPQA corpus consisting of 535 documents manually annotated for phrase-level expressions of opinion. In addition, the NIST-sponsored TREC evaluation has begun to develop data focusing on opinions — the 2003 Novelty Track features a task that requires systems to identify opinion-oriented documents w.r.t. a specific issue (Voorhees and Buckland, 2003).

While all of the above work begins to bridge the gap between text categorization and question answering, none of the approaches have been employed or evaluated in the context of MPQA.

3 OpQA Corpus

To support our research in MPQA, we created the OpQA corpus of opinion and fact questions and answers. Additional details on the construction of the corpus as well as results of an interannotator agreement study can be found in Stoyanov et al. (2004).

3.1 Documents and Questions

The OpQA corpus consists of 98 documents that appeared in the world press between June 2001 and May 2002. All documents were taken from the aforementioned MPQA corpus (Wilson and Wiebe, 2003)[1] and are manually annotated with phrase-level opinion information, following the annotation scheme of Wiebe et al. (2005), which is briefly summarized in Section 5. The documents cover four general (and controversial) topics: President Bush's alternative to the Kyoto protocol (*kyoto*); the US annual human rights report (*humanrights*); the 2002 coup d'etat in Venezuela (*venezuela*); and the 2002 elections in Zimbabwe and Mugabe's reelection (*mugabe*). Each topic is covered by between 19 and 33 documents that were identified automatically via IR methods.

Both fact and opinion questions for each topic were added to the OpQA corpus by a volunteer not associated with the current project. The volunteer was provided with a set of instructions for creating questions together with two documents on each topic selected at random. He created between six and eight questions on each topic, evenly split between fact and opinion. The 30 questions are given in Table 1 sorted by topic.

3.2 Answer annotations

Answer annotations were added to the corpus by two annotators according to a set of annotation instruc-

[1]The MPQA corpus is available at `http://nrrc.mitre.org/NRRC/publications.htm`. The OpQA corpus is available upon request.

	Kyoto
1 f	What is the Kyoto Protocol about?
2 f	When was the Kyoto Protocol adopted?
3 f	Who is the president of the Kiko Network?
4 f	What is the Kiko Network?
5 o	Does the president of the Kiko Network approve of the US action concerning the Kyoto Protocol?
6 o	Are the Japanese unanimous in their opinion of Bush's position on the Kyoto Protocol?
7 o	How is Bush's decision not to ratify the Kyoto Protocol looked upon by Japan and other US allies?
8 o	How do European Union countries feel about the US opposition to the Kyoto protocol?
	Human Rights
1 f	What is the murder rate in the United States?
2 f	What country issues an annual report on human rights in the United States?
3 o	How do the Chinese regard the human rights record of the United States?
4 f	Who is Andrew Welsdan?
5 o	What factors influence the way in which the US regards the human rights records of other nations?
6 o	Is the US Annual Human Rights Report received with universal approval around the world?
	Venezuela
1 f	When did Hugo Chavez become President?
2 f	Did any prominent Americans plan to visit Venezuela immediately following the 2002 coup?
3 o	Did anything surprising happen when Hugo Chavez regained power in Venezuela after he was removed by a coup?
4 o	Did most Venezuelans support the 2002 coup?
5 f	Which governmental institutions in Venezuela were dissolved by the leaders of the 2002 coup?
6 o	How did ordinary Venezuelans feel about the 2002 coup and subsequent events?
7 o	Did America support the Venezuelan foreign policy followed by Chavez?
8 f	Who is Vice-President of Venezuela?
	Mugabe
1 o	What was the American and British reaction to the reelection of Mugabe?
2 f	Where did Mugabe vote in the 2002 presidential election?
3 f	At which primary school had Mugabe been expected to vote in the 2002 presidential election?
4 f	How long has Mugabe headed his country?
5 f	Who was expecting Mugabe at Mhofu School for the 2002 election?
6 o	What is the basis for the European Union and US critical attitude and adversarial action toward Mugabe?
7 o	What did South Africa want Mugabe to do after the 2002 election?
8 o	What is Mugabe's opinion about the West's attitude and actions towards the 2002 Zimbabwe election?

Table 1: Questions in the OpQA collection by topic. *f* in column 1 indicates a fact question; *o*, an opinion question.

tions.[2] Every text segment that *contributes* to an answer to any of the 30 questions is annotated as an answer. In particular, answer annotations include segments that constitute a *partial answer*. Partial answers either (1) lack the specificity needed to constitute a full answer (e.g., "before May 2004" partially answers the question *When was the Kyoto protocol ratified?* when a specific date is known) or (2) need to be combined with at least one additional answer segment to fully answer the question (e.g., the question *Are the Japanese unanimous in their opposition of Bush's position on the Kyoto protocol?* is answered only partially by a segment expressing a single opinion). In addition, annotators mark the minimum answer spans (e.g., "a Tokyo organization," vs. "a Tokyo organization representing about 150 Japanese groups").

4 Characteristics of opinion answers

Next, we use the OpQA corpus to analyze and compare the characteristics of fact vs. opinion questions. Based on our findings, we believe that QA systems based solely on traditional QA techniques are likely to be less effective at MPQA than they are at traditional fact-based QA.

4.1 Traditional QA architectures

Despite the wide variety of approaches implied by modern QA systems, almost all systems rely on the following two steps (subsystems), which have empirically proven to be effective:

- **IR module.** The QA system invokes an IR subsystem that employs traditional text similarity measures (e.g., tf/idf) to retrieve and rank document fragments (sentences or paragraphs) w.r.t. the question (query).

- **Linguistic filters.** QA systems employ a set of filters and text processing components to discard some document fragments. The following filters have empirically proven to be effective and are used universally:

  Semantic filters prefer an answer segment that matches the semantic class(es) associated with the question type (e.g., *date* or *time* for *when* questions; *person* or *organization* for *who* questions).

  Syntactic filters are also configured on the type of question. The most common and effective syntactic filters select a specific constituent (e.g., noun phrase) according to the question type (e.g., *who* question).

QA systems typically interleave the above two subsystems with a variety of different processing steps of both the question and the answer. The goal of the processing is to identify text fragments that contain an answer to the question. Typical QA systems do not perform any further text processing; they return the text fragment as it occurred in the text. [3]

4.2 Corpus-based analysis of opinion answers

We hypothesize that QA systems that conform to this traditional architecture will have difficulty handling opinion questions without non-trivial modification. In support of this hypothesis, we provide statistics from the OpQA corpus to illustrate some of the characteristics that distinguish answers to opinion vs. fact questions, and discuss their implications for a traditional QA system architecture.

Answer length. We see in Table 2 that the average length of opinion answers in the OpQA corpus

[2]The annotation instructions are available at `http://www.cs.cornell.edu/ ves/Publications/publications.htm`.

[3]This architecture is seen mainly in QA systems designed for TREC's "factoid" and "list" QA tracks. Systems competing in the relatively new "definition" or "other" tracks have begun to introduce new approaches. However, most such systems still rely on the IR step and return the text fragment as it occurred in the text.

	Number of answers	Length	Number of partials
fact	124	5.12	12 (9.68%)
opinion	415	9.24	154 (37.11%)

Table 2: Number of answers, average answer length (in tokens), and number of partial answers for fact/opinion questions.

is 9.24 tokens, almost double that of fact answers. Unfortunately, longer answers could present problems for some traditional QA systems. In particular, some of the more sophisticated algorithms that perform **additional processing** steps such as logical verifiers (Moldovan et al., 2002) may be less accurate or computationally infeasible for longer answers. More importantly, longer answers are likely to span more than a single syntactic constituent, rendering the syntactic filters, and very likely the semantic filters, less effective.

Partial answers. Table 2 also shows that over 37% of the opinion answers were marked as partial vs. 9.68% of the fact answers. The implications of partial answers for the traditional QA architecture are substantial: an MPQA system will require an **answer generator** to (1) distinguish between partial and full answers; (2) recognize redundant partial answers; (3) identify which subset of the partial answers, if any, constitutes a full answer; (4) determine whether additional documents need to be examined to find a complete answer; and (5) asemble the final answer from partial pieces of information.

Syntactic constituent of the answer. As discussed in Section 4.1, traditional QA systems rely heavily on the predicted syntactic and semantic class of the answer. Based on answer lengths, we speculated that opinion answers are unlikely to span a single constituent and/or semantic class. This speculation is confirmed by examining the phrase type associated with OpQA answers using Abney's (1996) CASS partial parser.[4] For each question, we count the number of times an answer segment for the question (in the manual annotations) matches each constituent type. We consider four constituent types – noun phrase (n), verb phrase (v), prepositional phrase (p), and clause (c) – and three matching criteria:

Fact						Opinion					
Question	# of answers	Matching Criteria			syn type	Question	# of answers	Matching Criteria			syn type
		ex	up	up/dn				ex	up	up/dn	
H 1	1	0	0	0		H 3	15	5	5	5	c
H 2	4	2	2	2	n	H 5	24	5	5	10	n
H 4	1	0	0	0		H 6	123	17	23	52	n
K 1	48	13	14	24	n	K 5	3	0	0	1	
K 2	38	13	13	19	n	K 6	34	6	5	12	c
K 3	1	1	1	1	c n	K 7	55	9	8	19	c
K 4	2	1	1	1	n	K 8	25	4	4	10	v
M 2	3	0	0	1		M 1	74	10	12	29	v
M 3	1	0	0	1		M 6	12	3	5	7	n
M 4	10	2	2	5	n	M 7	1	0	0	0	
M 5	3	1	1	2	c	M 8	3	0	0	1	
V 1	4	3	3	4	n	V 3	1	1	0	1	c
V 2	1	1	1	1	n	V 4	13	2	2	2	c
V 5	3	0	1	1		V 6	9	2	2	5	c n
V 8	4	2	4	4	n	V 7	23	3	1	5	
Coverage	124	39 31%	43 35%	66 53%		Coverage	415	67 16%	70 17%	159 38%	

Table 3: Syntactic Constituent Type for Answers in the OpQA Corpus

1. The **ex**act match criterion is satisfied only by answer segments whose spans exactly correspond to a constituent in the CASS output.

2. The **up** criterion considers an answer to match a CASS constituent if the constituent completely contains the answer and no more than three additional (non-answer) tokens.

3. The **up/dn** criterion considers an answer to match a CASS constituent if it matches according to the **up** criterion or if the answer completely contains the constituent and no more than three additional tokens.

The counts for the analysis of answer segment syntactic type for fact vs. opinion questions are summarized in Table 3. Results for the 15 fact questions are shown in the left half of the table, and for the 15 opinion questions in the right half. The leftmost column in each half provides the question topic and number, and the second column indicates the total number of answer segments annotated for the question. The next three columns show, for each of the **ex**, **up**, and **up/dn** matching criteria, respectively, the number of annotated answer segments that match the majority syntactic type among answer segments for that question/criterion pair. Using a traditional QA architecture, the MPQA system might filter answers based on this majority type. The *syn type* column indicates the majority syntactic type using the exact match criterion; two values in the column indicate a tie for majority syntactic type, and an empty syntactic type indicates that no answer exactly matched any of the four constituent types. With only a few exceptions, the **up** and **up/dn** matching criteria agreed in majority syntactic type.

Results in Table 3 show a significant disparity between fact and opinion questions. For fact ques-

[4]The parser is available from `http://www.vinartus.net/spa/`.

tions, the syntactic type filter would keep 31%, 35%, or 53% of the correct answers, depending on the matching criterion. For opinion questions, there is unfortunately a two-fold reduction in the percentage of correct answers that would remain after filtering — only 16%, 17% or 38%, depending on the matching criterion. More importantly, the majority syntactic type among answers for fact questions is almost always a noun phrase, while no single constituent type emerges as a useful syntactic filter for opinion questions (see the **syn phrase** columns in Table 3). Finally, because semantic class information is generally tied to a particular syntactic category, the effectiveness of traditional semantic filters in the MPQA setting is unclear.

In summary, identifying answers to questions in an MPQA setting within a traditional QA architecture will be difficult. First, the implicit and explicit assumptions inherent in standard linguistic filters are consistent with the characteristics of fact- rather than opinion-oriented QA. In addition, the presence of relatively long answers and partial answers will require a much more complex **answer generator** than is typically present in current QA systems.

In Sections 6 and 7, we propose initial steps towards modifying the traditional QA architecture for use in MPQA. In particular, we propose and evaluate two types of **opinion filters** for MPQA: **subjectivity filters** and **opinion source filters**. Both types of linguistic filters rely on phrase-level and sentence-level opinion information, which has been manually annotated for our corpus; the next section briefly describes the opinion annotation scheme.

5 Manual Opinion Annotations

Documents in our OpQA corpus come from the larger MPQA corpus, which contains manual opinion annotations. The annotation framework is described in detail in (Wiebe et al., 2005). Here we give a high-level overview.

The annotation framework provides a basis for *subjective expressions*: expressions used to express opinions, emotions, and sentiments. The framework allows for the annotation of both directly expressed private states (e.g., *afraid* in the sentence "John is afraid that Sue might fall,") and opinions expressed by the choice of words and style of language (e.g., *it is about time* and *oppression* in the sentence "It is about time that we end Saddam's oppression"). In addition, the annotations include several attributes, including the *intensity* (with possible values *low*, *medium*, *high*, and *extreme*) and the *source* of the private state. The *source* of a private state is the person or entity who holds or experiences it.

6 Subjectivity Filters for MPQA Systems

This section describes three **subjectivity filters** based on the above opinion annotation scheme. Below (in Section 6.3), the filters are used to remove fact sentences from consideration when answering opinion questions, and the OpQA corpus is used to evaluate their effectiveness.

6.1 Manual Subjectivity Filter

Much previous research on automatic extraction of opinion information performed classifications at the sentence level. Therefore, we define sentence-level opinion classifications in terms of the phrase-level annotations. For our gold standard of manual opinion classifications (dubbed MANUAL for the rest of the paper) we will follow Riloff and Wiebe's (2003) convention (also used by Wiebe and Riloff (2005)) and consider a sentence to be *opinion* if it contains at least one opinion of intensity *medium* or higher, and to be *fact* otherwise.

6.2 Two Automatic Subjectivity Filters

As discussed in section 2, several research efforts have attempted to perform automatic opinion classification on the clause and sentence level. We investigate whether such information can be useful for MPQA by using the automatic sentence level opinion classifiers of Riloff and Wiebe (2003) and Wiebe and Riloff (2005).

Riloff and Wiebe (2003) use a bootstrapping algorithm to perform a sentence-based opinion classification on the MPQA corpus. They use a set of high precision subjectivity and objectivity clues to identify subjective and objective sentences. This data is then used in an algorithm similar to AutoSlog-TS (Riloff, 1996) to automatically identify a set of extraction patterns. The acquired patterns are then used iteratively to identify a larger set of subjective and objective sentences. In our experiments we use

		precision	recall	F
MPQA corpus	RULEBASED	90.4	34.2	46.6
	NAIVE BAYES	79.4	70.6	74.7

Table 4: Precision, recall, and F-measure for the two classifiers.

the classifier that was created by the reimplementation of this bootstrapping process in Wiebe and Riloff (2005). We will use RULEBASED to denote the opinion information output by this classifier.

In addition, Wiebe and Riloff used the RULEBASED classifier to produce a labeled data set for training. They trained a Naive Bayes subjectivity classifier on the labeled set. We will use NAIVE BAYES to refer to Wiebe and Riloff's naive Bayes classifier.[5] Table 4 shows the performance of the two classifiers on the MPQA corpus as reported by Wiebe and Riloff.

6.3 Experiments

We performed two types of experiments using the subjectivity filters.

6.3.1 Answer rank experiments

Our hypothesis motivating the first type of experiment is that subjectivity filters can improve the answer identification phase of an MPQA system. We implement the IR subsystem of a traditional QA system, and apply the subjectivity filters to the IR results. Specifically, for each opinion question in the corpus [6] , we do the following:

1. Split all documents in our corpus into sentences.
2. Run an information retrieval algorithm[7] on the set of all sentences using the question as the query to obtain a *ranked list* of sentences.
3. Apply a subjectivity filter to the *ranked list* to remove all fact sentences from the *ranked list*.

We test each of the MANUAL, RULEBASED, and NAIVE BAYES subjectivity filters. We compare the rank of the first answer to each question in the

Topic	Qnum	Baseline	Manual	NaiveBayes	Rulebased
Kyoto	5	1	1	1	1
	6	5	4	4	3
	7	1	1	1	1
	8	1	1	1	1
Human Rights	3	1	1	1	1
	5	10	6	7	5
	6	1	1	1	1
Venezuela	3	106	81	92	35
	4	3	2	3	1
	6	1	1	1	1
	7	3	3	3	2
Mugabe	1	2	2	2	2
	6	7	5	5	4
	7	447	291	317	153
	8	331	205	217	182
MRR :		0.4244	0.5189	0.5078	0.5856
MRFA:		61.3333	40.3333	43.7333	26.2

Table 5: Results for the subjectivity filters.

ranked list before the filter is applied, with the rank of the first answer to the question in the *ranked list* after the filter is applied.

Results. Results for the opinion filters are compared to a simple baseline, which performs the information retrieval step with no filtering. Table 5 gives the results on the 15 opinion questions for the baseline and each of the three *subjectivity filters*. The table shows two cumulative measures – the mean reciprocal rank (MRR) [8] and the mean rank of the first answer (MRFA). [9]

Table 5 shows that all three *subjectivity filters* outperform the baseline: for all three filters, the first answer in the filtered results for all 15 questions is ranked at least as high as in the baseline. As a result, the three subjectivity filters outperform the baseline in both MRR and MRFA. Surprisingly, the best performing subjectivity filter is RULEBASED, surpassing the gold standard MANUAL, both in MRR (0.59 vs. 0.52) and MRFA (40.3 vs. 26.2). Presumably, the improvement in performance comes from the fact that RULEBASED identifies subjective sentences with the highest precision (and lowest recall). Thus, the RULEBASED subjectivity filter discards non-subjective sentences most aggressively.

6.3.2 Answer probability experiments

The second experiment, *answer probability*, begins to explore whether opinion information can be

[5]Specifically, the one they label *Naive Bayes 1*.

[6]We do not evaluate the opinion filters on the 15 fact questions. Since opinion sentences are defined as containing at least one opinion of intensity medium or higher, opinion sentences can contain factual information and sentence-level opinion filters are not likely to be effective for fact-based QA.

[7]We use the Lemur toolkit's standard tf.idf implementation available from `http://www.lemurproject.org/`.

[8]The MRR is computed as the average of $1/r$, where r is the rank of the first answer.

[9]MRR has been accepted as the standard performance measure in QA, since MRFA can be strongly affected by outlier questions. However, the MRR score is dominated by the results in the high end of the ranking. Thus, MRFA may be more appropriate for our experiments because the filters are an intermediate step in the processing, the results of which other MPQA components may improve.

			sentence	
			fact	opinion
question	Manual	fact	56 (46.67%)	64 (53.33%)
		opinion	42 (10.14%)	372 (89.86%)
	Naive Bayes	fact	49 (40.83%)	71 (59.17%)
		opinion	57 (13.77%)	357 (86.23%)
	Rulebased	fact	96 (80.00%)	24 (20.00%)
		opinion	184 (44.44%)	230 (55.56%)

Table 6: Answer probability results.

used in an **answer generator**. This experiment considers correspondences between (1) the classes (i.e., opinion or fact) assigned by the subjectivity filters to the sentences containing answers, and (2) the classes of the questions the answers are responses to (according to the OpQA annotations). That is, we compute the probabilities (where *ans* = answer):
P(*ans* is in a *C1* sentence | *ans* is the answer to a *C2* question) for all four combinations of *C1*=opinion, fact and *C2*=opinion, fact.

Results. Results for the answer probability experiment are given in Table 6. The rows correspond to the classes of the questions the answers respond to, and the columns correspond to the classes assigned by the subjectivity filters to the sentences containing the answers. The first two rows, for instance, give the results for the MANUAL criterion. MANUAL placed 56 of the answers to fact questions in fact sentences (46.67% of all answers to fact questions) and 64 (53.33%) of the answers to fact questions in opinion sentences. Similarly, MANUAL placed 42 (10.14%) of the answers to opinion questions in fact sentences, and 372 (89.86%) of the answers to opinion questions in opinion sentences.

The answer probability experiment sheds some light on the subjectivity filter experiments. All three subjectivity filters place a larger percentage of answers to opinion questions in opinion sentences than they place in fact sentences. However, the different filters exhibit different degrees of discrimination. Answers to opinion questions are almost always placed in opinion sentences by MANUAL (89.86%) and NAIVE BAYES (86.23%). While that aspect of their performance is excellent, MANUAL and NAIVE BAYES place more answers to fact questions in opinion rather than fact sentences (though the percentages are in the 50s). This is to be expected, because MANUAL and NAIVE BAYES are more conservative and err on the side of classifying sentences as opinions: for MANUAL, the presence of any subjective expression makes the entire sentence opinion, even if parts of the sentence are factual; NAIVE BAYES shows high recall but lower precision in recognizing opinion sentences (see Table 4). Conversely, RULEBASED places 80% of the fact answers in fact sentences and only 56% of the opinion answers in opinion sentences. Again, the lower number of assignments to opinion sentences is to be expected, given the high precision and low recall of the classifier. But the net result is that, for RULEBASED, the off-diagonals are all less than 50%: it places more answers to fact questions in fact rather than opinion sentences (80%), and more answers to opinion questions in opinion rather than fact sentences (56%). This is consistent with its superior performance in the subjectivity filtering experiment.

In addition to explaining the performance of the subjectivity filters, the answer rank experiment shows that the automatic opinion classifiers can be used directly in an **answer generator** module. The two automatic classifiers rely on evidence in the sentence to predict the class (the information extraction patterns used by RULEBASED and the features used by NAIVE BAYES). In ongoing work we investigate ways to use this evidence to extract and summarize the opinions expressed in text, which is a task similar to that of an **answer generator** module.

7 Opinion Source Filters for MPQA Systems

In addition to subjectivity filters, we also define an opinion *source filter* based on the manual opinion annotations. This filter removes all sentences that do not have an opinion annotation with a source that matches the source of the question[10]. For this filter we only used the MANUAL source annotations since we did not have access to automatically extracted source information. We employ the same Answer Rank experiment as in 6.3.1, substituting the source filter for a subjectivity filter.

Results. Results for the source filter are mixed. The filter outperforms the baseline on some questions and performs worst on others. As a result the MRR for the source filter is worse than the base-

[10]We manually identified the sources of each of the 15 opinion questions.

line (0.4633 vs. 0.4244). However, the source filter exhibits by far the best results using the MRFA measure, a value of 11.267. The performance improvement is due to the filter's ability to recognize the answers to the hardest questions, for which the other filters have the most trouble (questions *mugabe* 7 and 8). For these questions, the rank of the first answer improves from 153 to 21, and from 182 to 11, respectively. With the exception of question *venezuela* 3, which does not contain a clear source (and is problematic altogether because there is only a single answer in the corpus and the question's qualification as opinion is not clear) the *source filter* always ranked an answer within the first 25 answers. Thus, *source filters* can be especially useful in systems that rely on the presence of an answer within the first few ranked answer segments and then invoke more sophisticated analysis in the **additional processing** phase.

8 Conclusions

We began by giving a high-level overview of the OpQA corpus. Using the corpus, we compared the characteristics of answers to fact and opinion questions. Based on the different characteristics, we surmise that traditional QA approaches may not be as effective for MPQA as they have been for fact-based QA. Finally, we investigated the use of machine learning and rule-based opinion filters and showed that they can be used to guide MPQA systems.

Acknowledgments We would like to thank Diane Litman for her work eliciting the questions for the OpQA corpus, and the anonymous reviewers for their helpful comments.This work was supported by the Advanced Research and Development Activity (ARDA), by NSF Grants IIS-0208028 and IIS-0208798, by the Xerox Foundation, and by a NSF Graduate Research Fellowship to the first author.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

S. Bethard, H. Yu, A. Thornton, V. Hativassiloglou, and D. Jurafsky. 2004. Automatic extraction of opinion propositions and their holders. In *2004 AAAI Spring Symposium on Exploring Attitude and Affect in Text*.

S. Das and M. Chen. 2001. Yahoo for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the 8th Asia Pacific Finance Association Annual Conference*.

Kushal Dave, Steve Lawrence, and David Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *International World Wide Web Conference*, pages 519–528.

D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. 2002. LCC tools for question answering. In *Proceedings of TREC 2002*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceesings of EMNLP*.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. *Proceedings of AAAI*.

V. Stoyanov, C. Cardie, J. Wiebe, and D. Litman. 2004. Evaluating an opinion annotation scheme using a new Multi-Perspective Question and Answer corpus. In *2004 AAAI Spring Symposium on Exploring Attitude and Affect in Text*.

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the ACL*, pages 417–424.

E. Voorhees and L. Buckland. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of TREC 12*.

Ellen Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *Proceedings of TREC 10*.

Ellen Voorhees. 2002. Overview of the 2002 Question Answering Track. In *Proceedings of TREC 11*.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of CICLing*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 1(2).

Theresa Wilson and Janyce Wiebe. 2003. Annotating opinions in the world press. *4th SIGdial Workshop on Discourse and Dialogue (SIGdial-03)*.

T. Wilson, J. Wiebe, and R. Hwa. 2004. Just how mad are you? Finding strong and weak opinion clauses. In *Proceedings of AAAI*.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*.

Automatically Evaluating Answers to Definition Questions

Jimmy Lin[1,3] **and Dina Demner-Fushman**[2,3]
[1]College of Information Studies
[2]Department of Computer Science
[3]Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742, USA
jimmylin@umd.edu, demner@cs.umd.edu

Abstract

Following recent developments in the automatic evaluation of machine translation and document summarization, we present a similar approach, implemented in a measure called POURPRE, for automatically evaluating answers to definition questions. Until now, the only way to assess the correctness of answers to such questions involves manual determination of whether an information nugget appears in a system's response. The lack of automatic methods for scoring system output is an impediment to progress in the field, which we address with this work. Experiments with the TREC 2003 and TREC 2004 QA tracks indicate that rankings produced by our metric correlate highly with official rankings, and that POURPRE outperforms direct application of existing metrics.

1 Introduction

Recent interest in question answering has shifted away from factoid questions such as "What city is the home to the Rock and Roll Hall of Fame?", which can typically be answered by a short noun phrase, to more complex and difficult questions. One interesting class of information needs concerns so-called definition questions such as "Who is Vlad the Impaler?", whose answers would include "nuggets" of information about the 16th century warrior prince's life, accomplishments, and legacy. Actually a misnomer, definition questions can be better paraphrased as "Tell me interesting things about *X*.", where *X* can be a person, an organization, a common noun, etc. Taken another way, definition questions might be viewed as simultaneously asking a whole series of factoid questions about the same entity (e.g., "When was he born?", "What was his occupation?", "Where did he live?", etc.), except that these questions are not known in advance; see Prager et al. (2004) for an implementation based on this view of definition questions.

Much progress in natural language processing and information retrieval has been driven by the creation of reusable test collections. A test collection consists of a corpus, a series of well-defined tasks, and a set of judgments indicating the "correct answers". To complete the picture, there must exist meaningful metrics to evaluate progress, and ideally, a machine should be able to compute these values automatically. Although "answers" to definition questions are known, there is no way to automatically and objectively determine if they are present in a given system's response (we will discuss why in Section 2). The experimental cycle is thus tortuously long; to accurately assess the performance of new techniques, one must essentially wait for expensive, large-scale evaluations that employ human assessors to judge the runs (e.g., the TREC QA track). This situation mirrors the state of machine translation and document summarization research a few years ago. Since then, however, automatic scoring metrics such as BLEU and ROUGE have been introduced as stopgap measures to facilitate experimentation.

Following these recent developments in evalua-

1	*vital*	32 kilograms plutonium powered
2	*vital*	seven year journey
3	*vital*	Titan 4-B Rocket
4	*vital*	send Huygens to probe atmosphere of Titan, Saturn's largest moon
5	*okay*	parachute instruments to planet's surface
6	*okay*	oceans of ethane or other hydrocarbons, frozen methane or water
7	*vital*	carries 12 packages scientific instruments and a probe
8	*okay*	NASA primary responsible for Cassini orbiter
9	*vital*	explore remote planet and its rings and moons, Saturn
10	*okay*	European Space Agency ESA responsible for Huygens probe
11	*okay*	controversy, protest, launch failure, re-entry, lethal risk, humans, plutonium
12	*okay*	Radioisotope Thermoelectric Generators, RTG
13	*vital*	Cassini, NASA'S Biggest and most complex interplanetary probe
14	*okay*	find information on solar system formation
15	*okay*	Cassini Joint Project between NASA, ESA, and ASI (Italian Space Agency)
16	*vital*	four year study mission

Table 1: The "answer key" to the question "What is the Cassini space probe?"

tion research, we propose POURPRE, a technique for automatically evaluating answers to definition questions. Like the abovementioned metrics, POURPRE is based on *n*-gram co-occurrences, but has been adapted for the unique characteristics of the question answering task. This paper will show that POURPRE can accurately assess the quality of answers to definition questions without human intervention, allowing experiments to be performed with rapid turnaround. We hope that this will enable faster exploration of the solution space and lead to accelerated advances in the state of the art.

This paper is organized as follows: In Section 2, we briefly describe how definition questions are currently evaluated, drawing attention to many of the intricacies involved. We discuss previous work in Section 3, relating POURPRE to evaluation metrics for other language applications. Section 4 discusses metrics for evaluating the quality of an automatic scoring algorithm. The POURPRE measure itself is outlined in Section 5; POURPRE scores are correlated with official human-generated scores in Section 6, and also compared to existing metrics. In Section 7, we explore the effect that judgment variability has on the stability of definition question evaluation, and its implications for automatic scoring algorithms.

2 Evaluating Definition Questions

To date, NIST has conducted two formal evaluations of definition questions, at TREC 2003 and TREC 2004.[1] In this section, we describe the setup of the task and the evaluation methodology.

Answers to definition questions are comprised of an unordered set of [document-id, answer string] pairs, where the strings are presumed to provide some relevant information about the entity being "defined", usually called the target. Although no explicit limit is placed on the length of the answer string, the final scoring metric penalizes verbosity (discussed below).

To evaluate system responses, NIST pools answer strings from all systems, removes their association with the runs that produced them, and presents them to a human assessor. Using these responses and research performed during the original development of the question, the assessor creates an "answer key"—a list of "information nuggets" about the target. An information nugget is defined as a fact for which the assessor could make a binary decision as to whether a response contained that nugget (Voorhees, 2003). The assessor also manually classifies each nugget as

[1]TREC 2004 questions were arranged around "topics"; definition questions were implicit in the "other" questions.

[XIE19971012.0112] The Cassini space probe, due to be launched from Cape Canaveral in Florida of the United States tomorrow, has a 32 kilogram plutonium fuel payload to power its seven year journey to Venus and Saturn.
Nuggets assigned: 1, 2

[NYT19990816.0266] Early in the Saturn visit, Cassini is to send a probe named Huygens into the smog-shrouded atmosphere of Titan, the planet's largest moon, and parachute instruments to its hidden surface to see if it holds oceans of ethane or other hydrocarbons over frozen layers of methane or water.
Nuggets assigned: 4, 5, 6

Figure 1: Examples of judging actual system responses.

either *vital* or *okay*. Vital nuggets represent concepts that must be present in a "good" definition; on the other hand, okay nuggets contribute worthwhile information about the target but are not essential; cf. (Hildebrandt et al., 2004). As an example, nuggets for the question "What is the Cassini space probe?" are shown in Table 1.

Once this answer key of vital/okay nuggets is created, the assessor then manually scores each run. For each system response, he or she decides whether or not each nugget is present. Assessors do not simply perform string matches in this decision process; rather, this matching occurs at the conceptual level, abstracting away from issues such as vocabulary differences, syntactic divergences, paraphrases, etc. Two examples of this matching process are shown in Figure 1: nuggets 1 and 2 were found in the top passage, while nuggets 4, 5, and 6 were found in the bottom passage. It is exactly this process of conceptually matching nuggets from the answer key with system responses that we attempt to capture with an automatic scoring algorithm.

The final F-score for an answer is calculated in the manner described in Figure 2, and the final score of a run is simply the average across the scores of all questions. The metric is a harmonic mean between nugget precision and nugget recall, where recall is heavily favored (controlled by the β parameter, set to five in 2003 and three in 2004). Nugget recall is calculated solely on vital nuggets, while nugget precision is approximated by a length allowance given based on the number of both vital and okay nuggets returned. Early on in a pilot study, researchers discovered that it was impossible for assessors to consistently enumerate the total set of nuggets contained

Let

r	# of *vital* nuggets returned in a response
a	# of *okay* nuggets returned in a response
R	# of *vital* nuggets in the answer key
l	# of non-whitespace characters in the entire answer string

Then

$$\text{recall } (\mathcal{R}) = r/R$$

$$\text{allowance } (\alpha) = 100 \times (r+a)$$

$$\text{precision } (\mathcal{P}) = \begin{cases} 1 & \text{if } l < \alpha \\ 1 - \frac{l-\alpha}{l} & \text{otherwise} \end{cases}$$

Finally, the $F(\beta) = \dfrac{(\beta^2+1) \times \mathcal{P} \times \mathcal{R}}{\beta^2 \times \mathcal{P} + \mathcal{R}}$

$\beta = 5$ in TREC 2003, $\beta = 3$ in TREC 2004.

Figure 2: Official definition of F-measure.

in a system response, given that they were usually extracted text fragments from documents (Voorhees, 2003). Thus, a penalty for verbosity serves as a surrogate for precision.

3 Previous Work

The idea of employing *n*-gram co-occurrence statistics to score the output of a computer system against one or more desired reference outputs was first successfully implemented in the BLEU metric for machine translation (Papineni et al., 2002). Since then, the basic method for scoring translation quality has been improved upon by others, e.g., (Babych and Hartley, 2004; Lin and Och, 2004). The basic idea has been extended to evaluating document summarization with ROUGE (Lin and Hovy, 2003).

Recently, Soricut and Brill (2004) employed *n*-gram co-occurrences to evaluate question answering in a FAQ domain; unfortunately, the task differs from definition question answering, making their results not directly applicable. Xu et al. (2004) applied ROUGE to automatically evaluate answers to definition questions, viewing the task as a variation of document summarization. Because TREC answer nuggets were terse phrases, the authors found it necessary to rephrase them—two humans were asked to manually create "reference answers" based on the assessors' nuggets and IR results, which was a labor-intensive process. Furthermore, Xu et al. did not perform a large-scale assessment of the reliability of ROUGE for evaluating definition answers.

4 Criteria for Success

Before proceeding to our description of POURPRE, it is important to first define the basis for assessing the quality of an automatic evaluation algorithm. Correlation between official scores and automatically-generated scores, as measured by the coefficient of determination R^2, seems like an obvious metric for quantifying the performance of a scoring algorithm. Indeed, this measure has been employed in the evaluation of BLEU, ROUGE, and other related metrics.

However, we believe that there are better measures of performance. In comparative evaluations, we ultimately want to determine if one technique is "better" than another. Thus, the system rankings produced by a particular scoring method are often more important than the actual scores themselves. Following the information retrieval literature, we employ Kendall's τ to capture this insight. Kendall's τ computes the "distance" between two rankings as the minimum number of pairwise adjacent swaps necessary to convert one ranking into the other. This value is normalized by the number of items being ranked such that two identical rankings produce a correlation of 1.0; the correlation between a ranking and its perfect inverse is -1.0; and the expected correlation of two rankings chosen at random is 0.0. Typically, a value of greater than 0.8 is considered "good", although 0.9 represents a threshold researchers generally aim for. In this study, we primarily focus on Kendall's τ, but also report R^2 values where appropriate.

5 POURPRE

Previously, it has been assumed that matching nuggets from the assessors' answer key with systems' responses must be performed manually because it involves semantics (Voorhees, 2003). We would like to challenge this assumption and hypothesize that term co-occurrence statistics can serve as a surrogate for this semantic matching process. Experience with the ROUGE metric has demonstrated the effectiveness of matching unigrams, an idea we employ in our POURPRE metric. We hypothesize that matching bigrams, trigrams, or any other longer *n*-grams will not be beneficial, because they primarily account for the fluency of a response, more relevant in a machine translation task. Since answers to definition questions are usually document extracts, fluency is less important a concern.

The idea behind POURPRE is relatively straightforward: match nuggets by summing the unigram co-occurrences between terms from each nugget and terms from the system response. We decided to start with the simplest possible approach: count the word overlap and divide by the total number of terms in the answer nugget. The only additional wrinkle is to ensure that all words appear within the same answer string. Since nuggets represent coherent concepts, they are unlikely to be spread across different answer strings (which are usually different extracts of source documents). As a simple example, let's say we're trying to determine if the nugget "A B C D" is contained in the following system response:

1. A
2. B C D
3. D
4. A D

The match score assigned to this nugget would be $3/4$, from answer string 2; no other answer string would get credit for this nugget. This provision reduces the impact of coincidental term matches.

Once we determine the match score for every nugget, the final F-score is calculated in the usual way, except that the automatically-derived match scores are substituted where appropriate. For example, nugget recall now becomes the sum of the match scores for all vital nuggets divided by the total number of vital nuggets. In the official F-score calcula-

Run	POURPRE				ROUGE	
	micro, cnt	macro, cnt	micro, *idf*	macro, *idf*	+stop	−stop
TREC 2004 ($\beta = 3$)	0.785	**0.833**	0.806	0.812	0.780	0.786
TREC 2003 ($\beta = 3$)	0.846	**0.886**	0.848	0.876	0.780	0.816
TREC 2003 ($\beta = 5$)	0.890	**0.878**	0.859	0.875	0.807	0.843

Table 2: Correlation (Kendall's τ) between rankings generated by POURPRE/ROUGE and official scores.

Run	POURPRE				ROUGE	
	micro, cnt	macro, cnt	micro, *idf*	macro, *idf*	+stop	−stop
TREC 2004 ($\beta = 3$)	0.837	**0.929**	0.904	0.914	0.854	0.871
TREC 2003 ($\beta = 3$)	0.919	**0.963**	0.941	0.957	0.876	0.887
TREC 2003 ($\beta = 5$)	0.954	**0.965**	0.957	0.964	0.919	0.929

Table 3: Correlation (R^2) between values generated by POURPRE/ROUGE and official scores.

tion, the length allowance—for the purposes of computing nugget precision—was 100 non-whitespace characters for every okay and vital nugget returned. Since nugget match scores are now fractional, this required some adjustment. We settled on an allowance of 100 non-whitespace characters for every nugget match that had non-zero score.

A major drawback of this basic unigram overlap approach is that all terms are considered equally important—surely, matching "year" in a system's response should count for less than matching "Huygens", in the example about the Cassini space probe. We decided to capture this intuition using inverse document frequency, a commonly-used measure in information retrieval; $idf(t_i)$ is defined as $log(N/c_i)$, where N is the number of documents in the collection, and c_i is the number of documents that contain the term t_i. With scoring based on *idf*, term counts are simply replaced with *idf* sums in computing the match score, i.e., the match score of a particular nugget is the sum of the *idf*s of matching terms in the system response divided by the sum of all term *idf*s from the answer nugget. Finally, we examined the effects of stemming, i.e., matching stemmed terms derived from the Porter stemmer.

In the next section, results of experiments with submissions to TREC 2003 and TREC 2004 are reported. We attempted two different methods for aggregating results: microaveraging and macroaveraging. For microaveraging, scores were calculated by computing the nugget match scores over all nuggets for all questions. For macroaveraging, scores for each question were first computed, and then averaged across all questions in the testset. With microaveraging, each nugget is given equal weight, while with macroaveraging, each question is given equal weight.

As a baseline, we revisited experiments by Xu et al. (2004) in using ROUGE to evaluate definition questions. What if we simply concatenated all the answer nuggets together and used the result as the "reference summary" (instead of using humans to create custom reference answers)?

6 Evaluation of POURPRE

We evaluated all definition question runs submitted to the TREC 2003[2] and TREC 2004 question answering tracks with different variants of our POURPRE metric, and then compared the results with the official F-scores generated by human assessors. The Kendall's τ correlations between rankings produced by POURPRE and the official rankings are shown in Table 2. The coefficients of determination (R^2) between the two sets of scores are shown in Table 3. We report four separate variants along two different parameters: scoring by term counts only vs. scoring by term *idf*, and microaveraging vs. macroaveraging. Interestingly, scoring based on macroaveraged term

[2]In TREC 2003, the value of β was arbitrarily set to five, which was later determined to favor recall too heavily. As a result, it was readjusted to three in TREC 2004. In our experiments with TREC 2003, we report figures for both values.

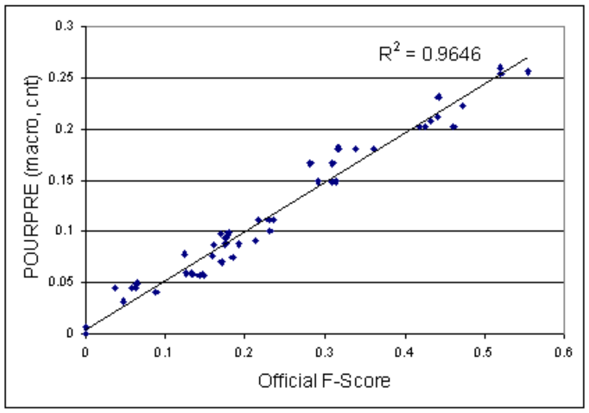


Figure 3: Scatter graph of official scores plotted against the POURPRE scores (macro, count) for TREC 2003 ($\beta = 5$).

counts outperformed any of the *idf* variants.

A scatter graph plotting official F-scores against POURPRE scores (macro, count) for TREC 2003 ($\beta = 5$) is shown in Figure 3. Corresponding graphs for other variants appear similar, and are not shown here. The effect of stemming on the Kendall's τ correlation between POURPRE (macro, count) and official scores in shown in Table 4. Results from the same stemming experiment on the other POURPRE variants are similarly inconclusive.

For TREC 2003 ($\beta = 5$), we performed an analysis of rank swaps between official and POURPRE scores. A rank swap is said to have occurred if the relative ranking of two runs is different under different conditions—they are significant because rank swaps might prevent researchers from confidently drawing conclusions about the relative effectiveness of different techniques. We observed 81 rank swaps (out of a total of 1431 pairwise comparisons for 54 runs). A histogram of these rank swaps, binned by the difference in official score, is shown in Figure 4. As can be seen, 48 rank swaps (59.3%) occurred when the difference in official score is less than 0.02; there were no rank swaps observed for runs in which the official scores differed by more than 0.061. Since measurement error is an inescapable fact of evaluation, we need not be concerned with rank swaps that can be attributed to this factor. For TREC 2003, Voorhees (2003) calculated this value to be approximately 0.1; that is, in order to conclude with 95% confidence that one run is better than another, an absolute F-score difference greater than 0.1 must be observed. As can be seen, all the rank swaps observed can be attributed to error inherent in the evaluation process.

Run	unstemmed	stemmed
TREC 2004 ($\beta = 3$)	0.833	0.825
TREC 2003 ($\beta = 3$)	0.886	0.897
TREC 2003 ($\beta = 5$)	0.878	0.895

Table 4: The effect of stemming on Kendall's τ; all runs with (macro, count) variant of POURPRE.

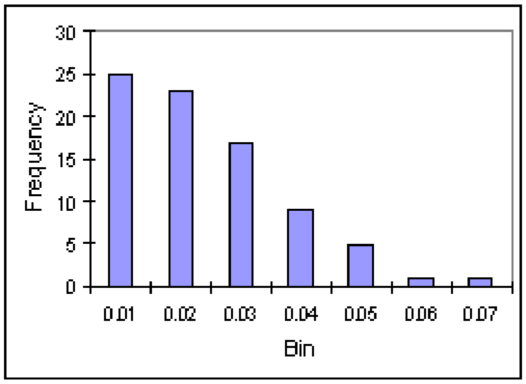


Figure 4: Histogram of rank swaps for TREC 2003 ($\beta = 5$), binned by difference in official score.

From these results, we can see that evaluation of definition questions is relatively coarse-grained. However, TREC 2003 was the first formal evaluation of definition questions; as methodologies are refined, the margin of error should go down. Although a similar error analysis for TREC 2004 has not been performed, we expect a similar result.

Given the simplicity of our POURPRE metric, the correlation between our automatically-derived scores and the official scores is remarkable. Starting from a set of questions and a list of relevant nuggets, POURPRE can accurately assess the performance of a definition question answering system without any human intervention.

6.1 Comparison Against ROUGE

We choose ROUGE over BLEU as a baseline for comparison because, conceptually, the task of answering definition questions is closer to summarization than it is to machine translation, in that both are recall-oriented. Since the majority of question answering systems employ extractive techniques, fluency (i.e., precision) is not usually an issue.

How does POURPRE stack up against using ROUGE[3] to directly evaluate definition questions? The Kendall's τ correlations between rankings produced by ROUGE (with and without stopword removal) and the official rankings are shown in Table 2; R^2 values are shown in Table 3. In all cases, ROUGE does not perform as well.

We believe that POURPRE better correlates with official scores because it takes into account special characteristics of the task: the distinction between vital and okay nuggets, the length penalty, etc. Other than a higher correlation, POURPRE offers an advantage over ROUGE in that it provides a better diagnostic than a coarse-grained score, i.e., it can reveal *why* an answer received a particular score. This allows researchers to conduct failure analyses to identify opportunities for improvement.

7 The Effect of Variability in Judgments

As with many other information retrieval tasks, legitimate differences in opinion about relevance are an inescapable fact of evaluating definition questions—systems are designed to satisfy real-world information needs, and users inevitably disagree on which nuggets are important or relevant. These disagreements manifest as scoring variations in an evaluation setting. The important issue, however, is the degree to which variations in judgments affect conclusions that can be drawn in a comparative evaluation, i.e., can we still confidently conclude that one system is "better" than another? For the *ad hoc* document retrieval task, research has shown that system rankings are stable with respect to disagreements about document relevance (Voorhees, 2000). In this section, we explore the effect of judgment variability on the stability and reliability of TREC definition question answering evaluations.

The vital/okay distinction on nuggets is one major source of differences in opinion, as has been pointed out previously (Hildebrandt et al., 2004). In the Cassini space probe example, we disagree with the assessors' assignment in many cases. More importantly, however, there does not appear to be any operationalizable rules for classifying nuggets as either vital or okay. Without any guiding principles, how can we expect our systems to automatically recognize this distinction?

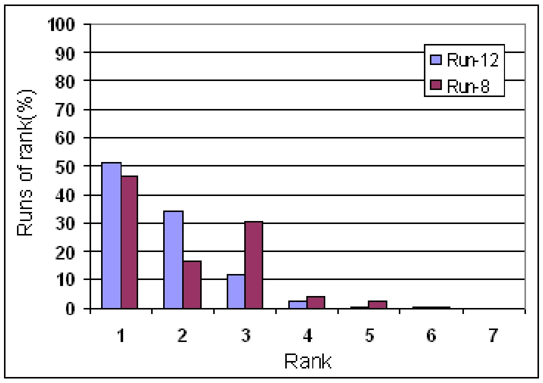


Figure 5: Distribution of rank placement using random judgments (for top two runs from TREC 2004).

How do differences in opinion about vital/okay nuggets impact the stability of system rankings? To answer this question, we measured the Kendall's τ correlation between the official rankings and rankings produced by different variations of the answer key. Three separate variants were considered:

- all nuggets considered vital
- vital/okay flipped (all vital nuggets become okay, and all okay nuggets become vital)
- randomly assigned vital/okay labels

Results are shown in Table 5. Note that this experiment was conducted with the manually-evaluated system responses, not our POURPRE metric. For the last condition, we conducted one thousand random trials, taking into consideration the original distribution of the vital and okay nuggets for each question using a simplified version of the Metropolis-Hastings algorithm (Chib and Greenberg, 1995); the standard deviations are reported.

These results suggest that system rankings are sensitive to assessors' opinion about what constitutes a vital or okay nugget. In general, the Kendall's τ values observed here are lower than values computed from corresponding experiments in *ad hoc* document retrieval (Voorhees, 2000). To illustrate, the distribution of ranks for the top two runs from

[3]We used ROUGE-1.4.2 with n set to 1, i.e. unigram matching, and maximum matching score rating.

Run	everything vital	vital/okay flipped	random judgments
TREC 2004 ($\beta = 3$)	0.919	0.859	0.841 $\pm$ 0.0195
TREC 2003 ($\beta = 3$)	0.927	0.802	0.822 $\pm$ 0.0215
TREC 2003 ($\beta = 5$)	0.920	0.796	0.808 $\pm$ 0.0219

Table 5: Correlation (Kendall's τ) between scores under different variations of judgments and the official scores. The 95% confidence interval is presented for the random judgments case.

TREC 2004 (RUN-12 and RUN-8) over the one thousand random trials is shown in Figure 5. In 511 trials, RUN-12 was ranked as the highest-scoring run; however, in 463 trials, RUN-8 was ranked as the highest-scoring run. Factoring in differences of opinion about the vital/okay distinction, one could not conclude with certainty which was the "best" run in the evaluation.

It appears that differences between POURPRE and the official scores are about the same as (or in some cases, smaller than) differences between the official scores and scores based on variant answer keys (with the exception of "everything vital"). This means that further refinement of the metric to increase correlation with human-generated scores may not be particularly meaningful; it might essentially amount to overtraining on the whims of a particular human assessor. We believe that sources of judgment variability and techniques for managing it represent important areas for future study.

8 Conclusion

We hope that POURPRE can accomplish for definition question answering what BLEU has done for machine translation, and ROUGE for document summarization: allow laboratory experiments to be conducted with rapid turnaround. A much shorter experimental cycle will allow researchers to explore different techniques and receive immediate feedback on their effectiveness. Hopefully, this will translate into rapid progress in the state of the art.[4]

9 Acknowledgements

This work was supported in part by ARDA's Advanced Question Answering for Intelligence (AQUAINT) Program. We would like to thank Donna Harman and Bonnie Dorr for comments on earlier drafts of this paper. In addition, we would like to thank Kiri for her kind support.

[4]A toolkit implementing the POURPRE metric can be downloaded at http://www.umiacs.umd.edu/~jimmylin/downloads/

References

Bogdan Babych and Anthony Hartley. 2004. Extending the BLEU MT evaluation method with frequency weightings. In *Proc. of ACL 2004*.

Siddhartha Chib and Edward Greenberg. 1995. Understanding the Metropolis-Hastings algorithm. *American Statistician*, 49(4):329–345.

Wesley Hildebrandt, Boris Katz, and Jimmy Lin. 2004. Answering definition questions with multiple knowledge sources. In *Proc. of HLT/NAACL 2004*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. of HLT/NAACL 2003*.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: A method for evaluating automatic evaluation metrics for machine translation. In *Proc. of COLING 2004*.

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

John Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. 2004. Question answering using constraint satisfaction: QA–by–Dossier–with–Constraints. In *Proc. of ACL 2004*.

Radu Soricut and Eric Brill. 2004. A unified framework for automatic evaluation using n-gram co-occurrence statistics. In *Proc. of ACL 2004*.

Ellen M. Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36(5):697–716.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Proc. of TREC 2003*.

Jinxi Xu, Ralph Weischedel, and Ana Licuanan. 2004. Evaluation of an extraction-based approach to answering definition questions. In *Proc. of SIGIR 2004*.

Integrating linguistic knowledge in passage retrieval for question answering

Jörg Tiedemann
Alfa Informatica, University of Groningen
Oude Kijk in 't Jatstraat 26
9712 EK Groningen, The Netherlands
j.tiedemann@rug.nl

Abstract

In this paper we investigate the use of linguistic knowledge in passage retrieval as part of an open-domain question answering system. We use annotation produced by a deep syntactic dependency parser for Dutch, Alpino, to extract various kinds of linguistic features and syntactic units to be included in a multi-layer index. Similar annotation is produced for natural language questions to be answered by the system. From this we extract query terms to be sent to the enriched retrieval index. We use a genetic algorithm to optimize the selection of features and syntactic units to be included in a query. This algorithm is also used to optimize further parameters such as keyword weights. The system is trained on questions from the competition on Dutch question answering within the Cross-Language Evaluation Forum (CLEF). We could show an improvement of about 15% in mean total reciprocal rank compared to traditional information retrieval using plain text keywords (including stemming and stop word removal).

1 Introduction

Improving information retrieval (IR) through natural language processing (NLP) has been the goal for many researchers. NLP techniques such as lemmatization and compound splitting have been used in several studies (Krovetz, 1993; Hollink et al., 2003). Linguistically motivated syntactic units such as noun phrases (Zhai, 1997), head-modifier pairs (Fagan, 1987; Strzalkowski et al., 1996) and subject-verb-object triples (Katz and Lin, 2003) have also been integrated in information retrieval. However, most of these studies resulted in only little success or even decreasing performance. It has been argued that NLP and especially deep syntactic analysis is still too brittle and ineffective (Katz and Lin, 2003).

Integrating NLP in information retrieval seems to be very hard because the task here is to match plain text keywords to natural language documents. In question answering (QA), however, the task is to match natural language questions to relevant answers within document collections. For this, we have to analyze the question in order to determine what kind of answer the user is expecting. Traditional information retrieval is used in QA systems to filter out relevant passages from the document collection which are then processed to extract possible answers. Hence, the performance of this *passage retrieval* component (especially in terms of recall) is crucial for the success of the entire system. NLP tools and linguistic resources are frequently used in QA systems, e.g. (Bernardi et al., 2003; Moldovan et al., 2002), although not very often for passage retrieval (some exceptions are (Strzalkowski et al., 1996; Katz and Lin, 2003; Neumann and Sacaleanu, 2004)).

Our goal is to utilize information that can be extracted from the analyzed question in order to match linguistic features and syntactic units in analyzed documents. The main research question is to find appropriate units and features that actually help to improve the retrieval component. Furthermore, we have to find an appropriate way of combining query terms to optimize IR performance. For this, we apply an iterative learning approach based on example questions annotated with their answers.

In the next section we will give a brief description of our question answering system with focus on the passage retrieval component. Thereafter we will discuss the query optimization algorithm followed by a section on experimental results. The final section contains our conclusions.

2 Question answering with dependency relations

Our Dutch question answering system, Joost (Bouma et al., 2005), consists of two streams: a table look-up strategy using off-line information extraction and an on-line strategy using passage retrieval and on-the-fly answer extraction. In both strategies we use syntactic information produced by a wide-coverage dependency parser for Dutch, Alpino (Bouma et al., 2001). In the off-line strategy we use syntactic patterns to extract information from unrestricted text to be stored in fact tables (Jijkoun et al., 2004). For the on-line strategy, we assume that there is a certain overlap between syntactic relations in the question and in passages containing the answers. Furthermore, we also use strategies for reasoning over dependency rules to capture semantic relationships that are expressed by different syntactic patterns (Bouma et al., 2005).

Our focus is set on open-domain question answering using data from the CLEF competition on Dutch QA. We have parsed the entire corpus provided by CLEF with about 4,000,000 sentences in about 190,000 documents. The dependency trees are stored in XML and are directly accessible from the QA system. Syntactic patterns for off-line information extraction are run on the entire corpus. For the on-line QA strategy we use traditional information retrieval to select relevant passages from the corpus to be processed by the answer extraction modules. This step is necessary to reduce the search space for the QA system to make it feasible to run on-line QA. As segmentation level we used paragraphs marked in the corpus (about 1.1 million).

Questions are parsed within the QA system using the same parser. Using their analysis, the system determines the question type and, hence, the expected answer type. According to the type, we try to find the answer first in the fact database (if an appropriate table exists) and then (as fallback) in the corpus using the on-line QA strategy.

2.1 Passage retrieval in Joost

Information retrieval is one of the bottle-necks in the on-line strategy of our QA system. The system relies on the passages retrieved by this component and fails if IR does not provide relevant documents. Traditional IR uses a bag-of-words approach using plain text keywords to be matched with word-vectors describing documents. The result is usually a ranked list of documents. Simple techniques such as stemming and stop word removal are used to improve the performance of such a system. This is also the base-line approach for passage retrieval in our QA system.

The passage retrieval component in Joost includes an interface to seven off-the shelf IR systems. One of the systems supported is Lucene from the Apache Jakarta project (Jakarta, 2004). Lucene is a widely-used open-source Java library with several extensions and useful features. This was the IR engine of our choice in the experiments described here. For the base-line we use standard settings and a public Dutch text analyzer for stemming and stop word removal. Now, the goal is to extend the base-line by incorporating linguistic information produced by the syntactic analyzer. Figure 1 shows a dependency tree produced for one of the sentences in the CLEF corpus. We like to include as much information from the parsed data as possible to find better matches between an analyzed question and passages that contain answers. From the parse trees, we extract various kinds of linguistic features and syntactic units to be stored in the index. Besides the dependency relations the parser also produces part-of-speech (POS) tags, named entity labels and linguistic root forms. It also recognizes compositional compounds and particle verbs. All this information might be useful for our passage retrieval component.

Lucene supports multiple index fields that can be filled with different types of data. This is a useful

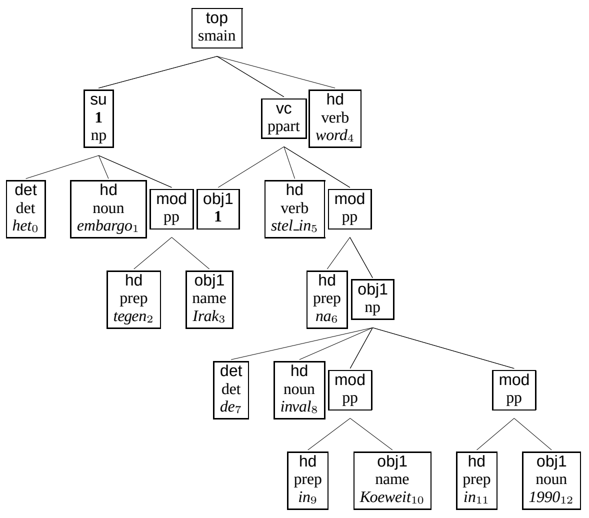


Figure 1: A dependency tree produced by Alpino: *Het embargo tegen Irak werd ingesteld na de inval in Koeweit in 1990.* (The embargo against Iraq has been declared after the invasion of Kuwait in 1990.)

feature since it allows one to store various kinds of information in different fields in the index. Henceforth, we will call these data fields *index layers* and, thus, the index will be called a *multi-layer index*. We distinguish between *token layers*, *type layers* and *annotation layers*. Token layers include one item per token in the corpus. Table 1 lists token layers defined in our index.

Table 1: Token layers

text	plain text tokens
root	root forms
RootPOS	root form + POS tag
RootHead	root form + head
RootRel	root form + relation name
RootRelHead	root form + relation + head

We included various combinations of features derived from the dependency trees to make it possible to test their impact on IR. Features are simply concatenated (using special delimiting symbols between the various parts) to create individual items in the layer. For example, the *RootHead* layer contains concatenated dependent-head bigrams taken from the dependency relations in the tree. Tokens in the *text* layer and in the *root* layer have been split at hyphens and underscores to split compositional compounds and particle verbs (Alpino adds underscores

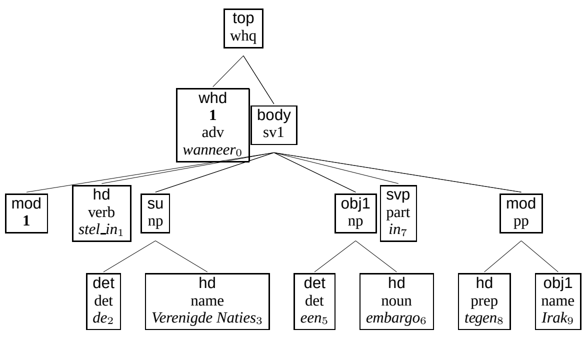


Figure 2: A dependency tree for a question: *Wanneer stelde de Verenigde Naties een embargo in tegen Irak?* (When did the United Nations declare the embargo against Iraq?)

between the compositional parts). Type layers include only specific types of tokens in the corpus, e.g. named entities or compounds (see table 2).

Table 2: Type layers

compound	compounds
ne	named entities
neLOC	location names
nePER	person names
neORG	organization names

Annotation layers include only the labels of (certain) token types. So far, we defined only one annotation layer for named entity labels. This layer may contain the items 'ORG', 'PER' or 'LOC' if such a named entity occurs in the text passage.

3 Query formulation

Questions are analyzed in the same way as sentences in documents. Hence, we can extract appropriate units from analyzed questions to be matched with the various layers in the index. For example, we can extract root-head word pairs to be matched with the RootHead layer. In this way, each layer can be queried using keywords of the same type. Furthermore, we can also use linguistic labels to restrict our query terms in several ways. For example, we can use part-of-speech labels to exclude keywords of a certain word class. We can also use the syntactic relation name to define query constraints. Each token layer can be restricted in this way (even if the feature used for the restriction is not part of the layer). For example, we can limit our set of root keywords to *nouns* even though part-of-speech labels are not part of the root layer. We can also combine constraints, for example, RootPOS keywords can be restricted to *nouns* that are in an *object* relation within the question.

Another feature of Lucene is the support of keyword weights. Keywords can be "boosted" using so-called "boost factors". Furthermore, keywords can also be marked as "required". These two features can be applied to all kinds of keywords (token layer, type layer, annotation layer keywords, and restricted keywords).

The following list summarizes possible keyword types in our passage retrieval component:

basic: a keyword in one of the index layers

restricted: *token-layer* keywords can be restricted to a certain word class and/or a certain relation type. We use only the following word class restrictions: *noun, name, adjective, verb*; and the following relation type restrictions: *direct object, modifier, apposition* and *subject*

weighted: keywords can be weighted using a *boost factor*

required: keywords can be marked as required

Query keywords from all types can be combined into a single query. We connect them in a disjunctive way which is the default operation in Lucene. The query engine provides ranked query results and, therefore, each disjunction may contribute to the ranking of the retrieved documents but does not harm the query if it does not produce any matching results. We may, for example, form a query with the following elements: (1) all plain *text* tokens; (2) named entities (*ne*) boosted with factor 2; (3) *RootHead* bigrams where the root is in an object relation; (4) *RootRel* keywords for all nouns. Applying these parameters to the question in figure 2 we get the following query:[1]

```
text:(Irak embargo Verenigde Naties stelde)
ne:(Irak^2 Verenigde_Naties^2)
RootHead:(Irak/tegen embargo/stel_in)
RootRel:(embargo/obj1)
```

Now, query terms from various keyword types may refer to the same index layer. For example, we may use weighted plain text keywords restricted to nouns together with unrestricted plain text keywords. To combine them we use a preference mechanism to keep queries simple and to avoid disjunctions with conflicting keyword parameters: (a) Restricted keyword types are more specific than basic keywords; (b) Keywords restricted in relation type *and* POS are more specific than keywords with only one restriction; (c) Relation type restrictions are more specific than POS restrictions. Using these rules we define that weights of more specific keywords overwrite weights of less specific ones. Furthermore, we define that the "required-marker" ('+') overwrites keyword weights. Using these definitions we would get the following query if we add two elements to the query from above: (5) plain text keywords in an object relation with boost factor 3 and (6) plain text keywords labeled as names marked as required.

```
text:(Irak^3 embargo^3 +Verenigde +Naties
stelde)
ne:(Irak^2 Verenigde_Naties^2)
RootHead:(Irak/tegen embargo/stel_in)
RootRel:(embargo/obj1)
```

Finally, we can also use the question type determined by question analysis in the retrieval component. The question type corresponds to the expected answer type, i.e. we expect an entity of that type in the relevant text passages. In some cases, the question type can be mapped to one of the named entity labels assigned by the parser, e.g. a *name question* is looking for names of persons (ne = PER), a question for a *capital* is looking for a location (ne = LOC) and a question for organizations is looking for the name of an organization (ne = ORG). Hence, we can add another keyword type, the expected answer type to be matched with named entity labels in the *ne* layer, cf. (Prager et al., 2000).

There are many possible combinations of restrictions even with the small set of POS labels and relation types listed above. However, many of them are useless because they cannot be instantiated. For example, an adjective cannot appear in subject relation to its head. For simplicity we limit ourselves to the following eight combined restrictions (POS + relation type): names + {direct object, modifier, apposition, subject} and nouns + {direct object, modifier, apposition, subject}. These can be applied to all token layers in the same way as the other restrictions using single constraints.

Altogether we have 109 different keyword types

[1]Note that stop words have been removed.

using the layers and the restrictions defined above. Now the question is to select appropriate keyword types among them with the optimal parameters (weights) to maximize retrieval performance. The following section describes the optimization procedure used to adjust query parameters.

4 Optimization of query parameters

In the previous sections we have seen the internal structure of the multi-layer index and the queries we use in our passage retrieval component. Now we have to address the question of how to select layers and restrict keywords to optimize the performance of the system according to the QA task. For this we employ an automatic optimization procedure that learns appropriate parameter settings from example data. We use annotated training material that is described in the next section. Thereafter, the optimization procedure is introduced.

4.1 CLEF questions and evaluation

We used results from the CLEF competition on Dutch QA from the years 2003 and 2004 for training and evaluation. They contain natural language questions annotated with their answers found in the CLEF corpus (answer strings and IDs of documents in which the answer was found). Most of the questions are factoid questions such as 'Hoeveel inwoners heeft Zweden?' (How many inhabitants does Sweden have?). Altogether there are 631 questions with 851 answers.[2]

Standard measures for evaluating information retrieval results are precision and recall. However, for QA several other specialized measures have been proposed, e.g. mean reciprocal rank (MRR) (Vorhees, 1999), coverage and redundancy (Roberts and Gaizauskas, 2004). MRR accounts only for the first passage retrieved containing an answer and disregards the following passages. Coverage and redundancy on the other hand disregard the ranking completely and focus on the sets of passages retrieved. However, in our QA system, the IR score (on which the retrieval ranking is based) is one of the clues used by the answer identification modules. Therefore, we use the *mean of the total reciprocal ranks* (MTRR), cf. (Radev et al., 2002), to combine features of all three measures:

$$MTRR = \frac{1}{x}\sum_{i=1}^{x}\sum_{d\in A_i}\frac{1}{rank_{R_i}(d)}$$

A_i is the set of retrieved passages containing an answer to question number i (subset of R_i) and $rank_{R_i}(d)$ is the rank of document d in the list of retrieved passages R_i.

In our experiments we used the provided answer string rather than the document ID to judge if a retrieved passage was relevant or not. In this way, the IR engine may provide passages with correct answers from other documents than the ones marked in the test set. We do simple string matching between answer strings and words in the retrieved passages. Obviously, this introduces errors where the matching string does not correspond to a valid answer in the context. However, we believe that this does not influence the global evaluation figure significantly and therefore we use this approach as a reasonable compromise when doing automatic evaluation.

4.2 Learning query parameters

As discussed earlier, there is a large variety of possible keyword types that can be combined to query the multi-layer index. Furthermore, we have a number of parameters to be set when formulating a query, e.g. the keyword weights. Selecting the appropriate keywords and parameters is not straightforward. We like to carry out a systematic search for optimizing parameters rather than using our intuition. Here, we use the information retrieval engine as a black box with certain input parameters. We do not know how the ranking is done internally or how the output is influenced by parameter changes. However, we can inspect and evaluate the output of the system. Hence, we need an iterative approach for testing several settings to optimize query parameters. The output for each setting has to be evaluated according to a certain objective function. For this, we need an automatic procedure because we want to check many different settings in a batch run. The performance of the system can be measured in several ways, e.g. us-

[2]Each question may have multiple possible answers. We also added some obvious answers which were not in the original test set when encountering them in the corpus. For example, names and numbers can be spelled differently (*Kim Jong Il* vs. *Kim Jong-Il, Saoedi-Arabië* vs. *Saudi-Arabië, bijna vijftig jaar* vs. *bijna 50 jaar*)

ing the MTRR scores described in the previous section. We have chosen to use this measure and the annotated CLEF questions to evaluate the retrieval performance automatically.

We decided to use a simplified genetic algorithm to optimize query parameters. This algorithm is implemented as an iterative "trial-and-error beam search" through possible parameter settings. The optimization loop works as follows (using a sub-set of the CLEF questions):

1. Run initial queries (one keyword type per IR run) with default weights.

2. Produce a number of new settings by combining two previous ones (= *crossover*). For this, select two settings from an N-best list from the previous IR runs. Apply *mutation* operations (see next step) until the new settings are unique (among all settings we have tried so far).

3. Change some of the new settings at random (= *mutation*) using pre-defi ned mutation operations.

4. Run the queries using the new settings and evaluate the retrieval output (determine *fitness*).

5. Continue with 2 until some stop condition is satisfi ed.

This optimization algorithm is very simple but requires some additional parameters. First of all, we have to set the size of the *population*, i.e. the number of IR runs (*individuals*) to be kept for the next iteration. We decided to keep the population small with only 25 individuals. Then we have to decide how to evaluate fitness to rank retrieval results. This is done using the MTRR measure. *Natural selection* using these rankings is simplified to a top-N search without giving individuals with lower fitness values a chance to survive. This also means that we can update the population directly when a new IR run is finished. We also have to set a maximum number of new settings to be created. In our experiments we limit the process to a maximum of 50 settings that may be tried simultaneously. A new setting is created as soon as there is a spot available.

An important part of the algorithm is the combination of parameters. We simply merge the settings of two previous runs (*parents*) to produce a new setting (a *child*). That means that all keyword types (with their restrictions) from both parents are included in the child's setting. Parents are selected at random without any preference mechanism. We also use a very simple strategy in cases where both parents contain the same keyword type. In these cases we compute the arithmetic mean of the weight assigned to this type in the parents' settings (default weight is one). If the keyword type is marked as required in one of the parents, it will also be marked as required in the child's setting (which will overwrite the keyword weight if it is set in the other parent).

Another important principle in genetic optimization is mutation. It refers to a randomized modification of settings when new individuals are created. First, we apply mutation operations where new settings are not unique.[3] Secondly, mutation operations are applied with fixed probabilities to new settings.

In most genetic algorithms, settings are converted to genes consisting of bit strings. A mutation operation is then defined as flipping the value of one randomly chosen bit. In our approach, we do not use bit strings but define several mutation operations to modify parameters directly. The following operations have been defined:

- a new keyword type is added to new settings with a chance of 0.2

- a keyword type is removed from the settings with a chance of 0.1

- a keyword weight (boost factor) is modified by a random value between -5 and 5 with a chance of 0.2 (but only if the weight remains a positive value)

- a keyword type is marked as required with a chance of 0.01

All these parameters are intuitively chosen. We assigned rather high probabilities to the mutation operations to reduce the risk of local maximum traps. Note that there is no obvious condition for termination. In randomized approaches like this one the development of the fitness score is most likely not monotonic and therefore, it is hard to predict when we should stop the optimization process. However, we expect the scores to converge at some point and we may stop if a certain number of new settings does not improve the scores anymore.

[3]We require unique settings in our implementation because we want to avoid re-computation of fi tness values for settings that have been tried already. "Good" settings survive anyway using our top-N selection approach.

5 Experiments

We selected a random set of 420 questions from the CLEF data for training and used the remaining 150 questions for evaluation. We used the optimization algorithm with the settings as described above. IR was run in parallel on 3-7 Linux workstations on a local network. We retrieved a maximum of 20 passages per question. For each setting we computed the fitness scores for the training set and the evaluation set using MTRR. The top scores have been printed after each 10 runs and compared to the evaluation scores. Figure 3 shows a plot of the fitness score development throughout the optimization process in comparison with the evaluation scores.

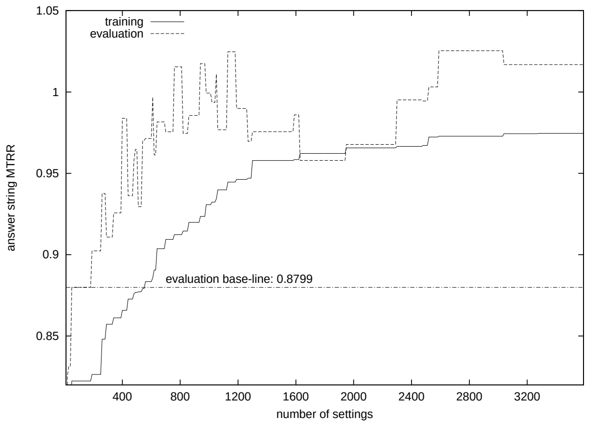


Figure 3: Parameter optimization

The base-line of 0.8799 refers to the retrieval result on evaluation data when using traditional IR with plain text keywords only (i.e. using the text layer, Dutch stemming and stop word removal). The base-line performance on training data is slightly worse with 0.8224 MTRR. After 1130 settings the MTRR scores increased to 0.9446 for training data and 1.0247 for evaluation data. Thereafter we can observe a surprising drop in evaluation scores to around 0.97 in MTRR. This might be due to overfitting although the drop seems to be rather radical. After that the curve of the evaluation scores goes back to about the same level as achieved before and the training curve seems to level out. The MTRR score after 3200 settings is at 1.0169 on evaluation data which is a statistically significant improvement of the baseline score (tested using the Wilcoxon matched-pairs signed-ranks test at $p < 0.01$). MTRR measured on document IDs and evaluation data did also increase from 0.5422 to 0.6215 which is statistically significant at p¡0.02. Coverage went up from 78.68% to 81.62% on evaluation data and the redundancy was improved from 3.824 to 4.272 (significance tests have not been carried out). Finally, the QA performance using Joost with only the IR based strategy was increased from 0.289 (using CLEF scores) to 0.331. This, however, is not statistically significant according to the Wilcoxon test and may be due to chance.

Table 3: Optimized parameters (3200 settings)

weighted keywords			required keywords	
layer	restriction	weight	layer	restriction
text		7.43	root	name
text	name	11.94		
text	adj	9.14	RootPOS	
text	mod	5.83	RootPOS	obj1
text	verb	4.33	RootPOS	noun-mod
text	noun-app	3.70		
root		4.45	RootRel	
root	noun-su	2.65	RootRel	app
root	name-mod	9.71	RootRel	noun-app
root	noun-obj1	0.09	RootRel	noun-mod
root	mod	0.81	RootRel	noun-obj1
root	verb	0.01		
RootHead	noun-app	7.65	RootRelHead	su
RootHead	noun-mod	5.24	RootRelHead	adj
RootHead	name-su	1	RootRelHead	name-app
RootRel	mod	4.45		
RootRel	name-app	2.17	Q-type	
RootRel	noun	2.49		
RootRelHead	obj1	1.60		
RootRelHead	name-su	1		
nePER		0.91		

Table 3 shows the features and weights selected in the training process. The largest weights are given to names in the text layer, to root forms of names in modifier relations and to plain text adjectives. Many keyword types use 'name' or 'noun' as POS restriction. A surprisingly large number of keyword types are marked as required. Some of them overlap with each other and are therefore redundant. For example, all RootPOS keywords are marked as required and therefore, the restrictions of RootPOS keywords are useless because they do not alter the query. However, in other cases overlapping keyword type definitions do influence the query. For example, RootRel keywords in general are marked as required. However, other type definitions replace some of them with weighted keywords, e.g., RootRel noun keywords. Finally, some of them may be changed back to required keywords, e.g., RootRel keywords of nouns in a modifier relation.

6 Conclusions

In this paper we describe an approach for integrating linguistic information derived from dependency analyses in passage retrieval for question answering. Our retrieval component uses a multi-layer index containing various combinations of linguistic features and syntactic units extracted from a fully analyzed corpus of unrestricted Dutch text. Natural language questions are parsed in the same way. Their analyses are used to build complex queries to our extended index. We demonstrated a genetic algorithm for optimizing query parameters to improve the retrieval performance. The system was trained on questions from the CLEF competition on open-domain question answering for Dutch which are annotated with corresponding answers in the corpus. We could show a significant improvement of about 15% in mean total reciprocal rank using extended queries with optimized parameters compared with the base-line of traditional information retrieval using plain text keywords.

References

Raffaella Bernardi, Valentin Jijkoun, Gilad Mishne, and Maarten de Rijke. 2003. Selectively using linguistic resources throughout the question answering pipeline. In *Proceedings of the 2nd CoLogNET-ElsNET Symposium*.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide coverage computational analysis of Dutch. In *Computational Linguistics in the Netherlands CLIN, 2000*. Rodopi.

Gosse Bouma, Jori Mur, and Gertjan van Noord. 2005. Reasoning over dependency relations for QA. In *Knowledge and Reasoning for Answering Questions (KRAQ'05)*, IJCAI Workshop, Edinburgh, Scotland.

Joel L. Fagan. 1987. Automatic phrase indexing for document retrieval. In *SIGIR '87: Proceedings of the 10th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–101, New York, NY, USA. ACM Press.

Vera Hollink, Jaap Kamps, Christof Monz, and Maarten de Rijke. 2003. Monolingual document retrieval for European languages. *Information Retrieval*, (6).

Apache Jakarta. 2004. Apache Lucene - a high-performance, full-featured text search engine library. http://lucene.apache.org/java/docs/index.html.

Valentin Jijkoun, Jori Mur, and Maarten de Rijke. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of COLING-2004*.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*.

Robert Krovetz. 1993. Viewing morphology as an inference process,. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–203.

Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. 2002. LCC tools for question answering. In *Proceedings of TREC-11*.

Günter Neumann and Bogdan Sacaleanu. 2004. Experiments on robust NL question interpretation and multi-layered document annotation for a cross-language question/answering system. In *Proceedings of the CLEF 2004 working notes of the QA@CLEF*, Bath.

John Prager, Eric Brown, Anni Cohen, Dragomir Radev, and Valerie Samn. 2000. Question-answering by predictive annotation. In *In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July.

Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating web-based question answering systems. In *Proceedings of LREC*, Las Palmas, Spain.

Ian Roberts and Robert Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *Proceedings of the 26th European Conference on Information Retrieval (ECIR)*, pages 72–84.

Tomek Strzalkowski, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, José Pérez-Carballo, Troy Straszheim, Jin Wang, and Jon Wilding. 1996. Natural language information retrieval: TREC-5 report.

Ellen M. Vorhees. 1999. The TREC-8 question answering track report. In *Proceedings of TREC-8*, pages 77–82.

Chengxiang Zhai. 1997. Fast statistical parsing of noun phrases for document indexing. In *Proceedings of the fi fth conference on Applied natural language processing*, pages 312–319, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

SEARCHING THE AUDIO NOTEBOOK: KEYWORD SEARCH IN RECORDED CONVERSATIONS

Peng Yu, Kaijiang Chen, Lie Lu, and **Frank Seide**
Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Rd., 100080 Beijing, P.R.C.
{rogeryu,kaijchen,llu,fseide}@microsoft.com

Abstract

MIT's Audio Notebook added great value to the note-taking process by retaining audio recordings, e.g. during lectures or interviews. The key was to provide users ways to quickly and easily access portions of interest in a recording. Several non-speech-recognition based techniques were employed. In this paper we present a system to search directly the audio recordings by key phrases. We have identified the user requirements as accurate ranking of phrase matches, domain independence, and reasonable response time. We address these requirements by a hybrid word/phoneme search in lattices, and a supporting indexing scheme. We will introduce the ranking criterion, a unified hybrid posterior-lattice representation, and the indexing algorithm for hybrid lattices. We present results for five different recording sets, including meetings, telephone conversations, and interviews. Our results show an average search accuracy of 84%, which is dramatically better than a direct search in speech recognition transcripts (less than 40% search accuracy).

1 Introduction

Lisa Stifelman proposed in her thesis the idea of the "Audio Notebook," where audio recordings of lectures and interviews are retained along with the notes (Stifelman, 1997). She has shown that the audio recordings are valuable to users if portions of interest can be accessed quickly and easily.

Stifelman explored various techniques for this, including user-activity based techniques (most noteworthy time-stamping notes so they can serve as an index into the recording) and content-based ones (signal processing for accelerated playback, "snap-to-grid" (=phrase boundary) based on prosodic cues). The latter are intended for situations where the former fail, e.g. when the user has no time for taking notes, does not wish to pay attention to it, or cannot keep up with complex subject matter, and as a consequence the audio is left without index. In this paper, we investigate technologies for searching the *spoken content* of the audio recording.

Several approaches have been reported in the literature for the problem of indexing spoken words in audio recordings. The TREC (Text REtrieval Conference) Spoken-Document Retrieval (SDR) track has fostered research on audio-retrieval of broadcast-news clips. Most TREC benchmarking systems use broadcast-news recognizers to generate approximate transcripts, and apply text-based information retrieval to these. They achieve retrieval accuracy similar to using human reference transcripts, and ad-hoc retrieval for broadcast news is considered a "solved problem" (Garofolo, 2000). Noteworthy are the rather low word-error rates (20%) in the TREC evaluations, and that recognition errors did not lead to catastrophic failures due to redundancy of news segments and queries.

However, in our scenario, requirements are rather different. First, word-error rates are much higher (40-60%). Directly searching such inaccurate speech recognition transcripts suffers from a poor recall. Second, unlike broadcast-news material, user recordings of conversations will not be limited to a few specific domains. This not only poses difficulties for obtaining domain-specific training data, but also implies an unlimited vocabulary of query phrases users want to use. Third, audio recordings will accumulate. When the audio database grows to hundreds or even thousands of hours, a reasonable response time is still needed.

A successful way to deal with high word error rates is the use of recognition alternates (lattices). For example, (Seide and Yu, 2004; Yu and Seide, 2004) reports a substantial 50% improvement of FOM (Figure Of Merit) for a word-spotting task in voicemails. Improvements from using lattices were also reported by (Saraclar and Sproat, 2004) and (Chelba and Acero, 2005).

To address the problem of domain independence, a subword-based approach is needed. In (Logan, 2002) the authors address the problem by indexing phonetic or word-fragment based transcriptions. Similar approaches, e.g. using overlapping M-grams of phonemes, are discussed in (Schäuble, 1995) and (Ng, 2000). (James and Young, 1994) introduces the approach of searching phoneme lattices. (Clements, 2001) proposes a similar idea called "phonetic search track." In previous work (Seide and Yu, 2004), promising results were obtained with phonetic lattice search in voicemails. In (Yu and

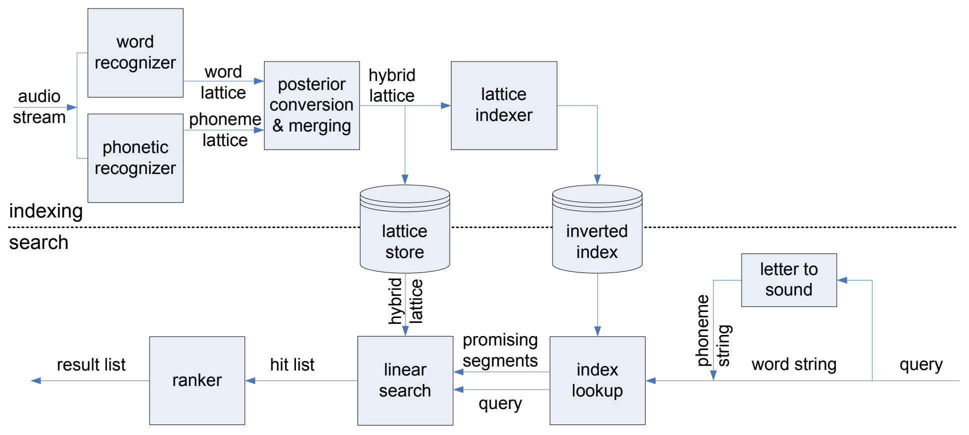


Figure 1: System architecture.

Seide, 2004), it was found that even better result can be achieved by combining a phonetic search with a word-level search.

For the third problem, quick response time is commonly achieved by indexing techniques. However, in the context of phonetic lattice search, the concept of "indexing" becomes a non-trivial problem, because due to the unknown-word nature, we need to deal with an open set of index keys. (Saraclar and Sproat, 2004) proposes to store the individual lattice arcs (inverting the lattice). (Allauzen *et al.*, 2004) introduces a general indexation framework by indexing expected term frequencies ("expected counts") instead of each individual keyword occurrence or lattice arcs. In (Yu *et al.*, 2005), a similar idea of indexing expected term frequencies is proposed, suggesting to approximate expected term frequencies by M-*gram phoneme language models* estimated on segments of audio.

In this paper, we combine previous work on phonetic lattice search, hybrid search and lattice indexing into a real system for searching recorded conversations that achieves high accuracy and can handle hundreds of hours of audio. The main contributions of this paper are: a real system for searching conversational speech, a novel method for combining phoneme and word lattices, and experimental results for searching recorded conversations.

The paper is organized as follows. Section 2 gives an overview of the system. Section 3 introduces the overall criterion, based on which the system is developed, Section 4 introduces our implementation for a hybrid word/phoneme search system, and Section 5 discusses the lattice indexing mechanism. Section 6 presents the experimental results, and Section 7 concludes.

2 A System For Searching Conversations

A system for searching the spoken content of recorded conversations has several distinct properties. Users are searching their own meetings, so most searches will be known-item searches with at most a few correct hits in the archive. Users will often search for specific phrases that they remember, possibly with boolean operators. Relevance weighting of individual query terms is less of an issue in this scenario.

We identified three user requirements:

- high recall and accurate ranking of phrase matches;
- domain independence – it should work for any topic, ideally without need to adapt vocabularies or language models;
- reasonable response time – a few seconds at most, independent of the size of the conversation archive.

We address them as follows. First, to increase recall we search *recognition alternates* based on *lattices*. Lattice oracle word-error rates[1] are significantly lower than word-error rates of the best path. For example, (Chelba and Acero, 2005) reports a lattice oracle error rate of 22% for lecture recordings at a top-1 word-error rate of 45%[2]. To utilize recognizer scores in the lattices, we formulating the ranking problem as one of risk minimization and derive that keyword hits should be ranked by their *word (phrase) posterior probabilities*.

Second, domain independence is achieved by combining large-vocabulary recognition with a phonetic search. This helps especially for proper names and specialized terminology, which are often either missing in the vocabulary or not well-predicted by the language model.

Third, to achieve quick response time, we use an M-gram based indexing approach. It has two stages, where the first stage is a fast index lookup to create a short-list of candidate lattices. In the second stage, a detailed lattice match is applied to the lattices in the short-list. We call the second stage *linear search* because search time grows linearly with the duration of the lattices searched.

[1]The "oracle word-error rate" of a lattice is the word error rate of the path through the lattice that has the least errors.

[2]Note that this comparison was for a reasonably well-tuned recognizer setup. Any arbitrary lattice oracle error rate can be obtained by adjusting the recognizer's pruning setup and investing enough computation time (plus possibly adapting the search-space organization).

The resulting system architecture is shown in Fig. 1. In the following three sections, we will discuss our solutions in these three aspects in details respectively.

3 Ranking Criterion

For ranking search results according to "relevance" to the user's query, several relevance measures have been proposed in the text-retrieval literature. The key element of these measures is weighting the contribution of individual keywords to the relevance-ranking score. Unfortunately, text ranking criteria are not directly applicable to retrieval of speech because recognition alternates and confidence scores are not considered.

Luckily, this is less of an issue in our known-item style search, because the simplest of relevance measures can be used: A search hit is assumed relevant if the query phrase was indeed said there (and fulfills optional boolean constraints), and it is not relevant otherwise.

This simple relevance measure, combined with a variant of the *probability ranking principle* (Robertson, 1977), leads to a system where phrase hits are ranked by their phrase posterior probability. This is derived through a Bayes-risk minimizing approach as follows:

1. Let the relevance be $R(Q, \mathrm{hit}^i)$ of a returned audio hit – hit^i to a user's query Q formally defined is 1 (match) if the hit is an occurrence of the query term with time boundaries $(t_s^{\mathrm{hit}^i}, t_e^{\mathrm{hit}^i})$, or 0 if not.

2. The user expects the system to return a list of audio hits, ranked such that the *accumulative relevance* of the top n hits $(\mathrm{hit}^1 ... \mathrm{hit}^n)$, averaged over a range of $n = 1...n_{\max}$, is maximal:
$$\frac{1}{n_{\max}} \sum_{n=1}^{n_{\max}} \sum_{i=1}^{n} R(Q, \mathrm{hit}^i) \quad \overset{!}{=} \quad \max . \qquad (1)$$
Note that this is closely related to popular word-spotting metrics, such as the NIST (National Institute of Standards & Technology) Figure Of Merit.

To the retrieval system, the true transcription of each audio file is unknown, so it must maximize Eq. (1) in the sense of an expected value

$$E_{WT|O}\left\{ \frac{1}{n_{\max}} \sum_{n=1}^{n_{\max}} \sum_{i=1}^{n} R_{WT}(Q, \mathrm{hit}^i) \right\} \quad \overset{!}{=} \quad \max,$$

where O denotes the totality of all audio files (O for observation), $W = (w_1, w_2, ..., w_N)$ a hypothesized transcription of the entire collection, and $T = (t_1, t_2, ..., t_{N+1})$ the associated time boundaries on a shared collection-wide time axis.

$R_{WT}(\cdot)$ shall be relevance w.r.t. the hypothesized transcription and alignment. The expected value is taken w.r.t. the posterior probability distribution $P(WT|O)$ provided by our speech recognizer in the form of scored lattices. It is easy to see that this expression is maximal if the hits are ranked by their expected relevance $E_{WT|O}\{R_{WT}(Q, \mathrm{hit}^i)\}$. In our definition of relevance, $R_{WT}(Q, \mathrm{hit}^i)$ is written as

$$R_{WT}(Q, \mathrm{hit}^i) \quad = \quad \begin{cases} 1 & \exists k, l : t_k = t_s^{\mathrm{hit}^i} \\ & \quad \wedge t_{k+l} = t_e^{\mathrm{hit}^i} \\ & \quad \wedge w_k, ..., w_{k+l-1} = Q \\ 0 & \text{otherwise} \end{cases}$$

and the expected relevance is computed as

$$\begin{aligned} E_{WT|O}\{R_{WT}(Q, \mathrm{hit}^i)\} &= \sum_{WT} R_{WT}(Q, \mathrm{hit}^i) P(WT|O) \\ &= P(*, t_s^{\mathrm{hit}^i}, Q, t_e^{\mathrm{hit}^i}, *|O) \end{aligned}$$

with

$$P(*, t_s, Q, t_e, *|O) \quad = \sum_{\substack{WT:\exists k,l:t_k=t_s \wedge t_{k+l}=t_e \\ \wedge w_k,...,w_{k+l-1}=Q}} P(WT|O). \qquad (2)$$

For single-word queries, this is the well-known *word posterior probability* (Wessel *et al.*, 2000; Evermann *et al.*, 2000). To cover multi-label phrase queries, we will call it *phrase posterior probability*.

The formalism in this section is applicable to all sorts of units, such as fragments, syllables, or words. The transcription W and its units w_k, as well as the query string Q, should be understood in this sense. For a regular word-level search, W and Q are just *strings of words* In the context of phonetic search, W and Q are *strings of phonemes*. For simplicity of notation, we have excluded the issue of multiple pronunciations of a word. Eq. (2) can be trivially extended by summing up over all alternative pronunciations of the query. And in a hybrid search, there would be multiple representations of the query, which are just as pronunciation variants.

4 Word/Phoneme Hybrid Search

For a combined word and phoneme based search, two problems need to be considered:

- Recognizer configuration. While established solutions exist for word-lattice generation, what needs to be done for generating high-quality phoneme lattices?
- How should word and phoneme lattices be jointly represented for the purpose of search, and how should they be searched?

4.1 Speech Recognition

4.1.1 Large-Vocabulary Recognition

Word lattices are generated by a common speaker-independent large-vocabulary recognizer. Because the speaking style of conversations is very different from, say, your average speech dictation system, specialized acoustic models are used. These are trained on conversational speech to match the speaking style. The vocabulary and the trigram language model are designed to cover a broad range of topics.

The drawback of large-vocabulary recognition is, of course, that it is infeasible to have the vocabulary cover all possible keywords that a user may use, particularly proper names and specialized terminology.

One way to address this *out-of-vocabulary problem* is to mine the user's documents or e-mails to adapt the recognizer's vocabulary. While this is workable for some scenarios, it is not a good solution e.g. when new words are frequently introduced in the conversations themselves rather than preceding written conversations, where the spelling of a new word is not obvious and thus inconsistent, or when documents with related documents are not easily available on the user's hard disk but would have to be specifically gathered by the user.

A second problem is that the performance of state-of-the-art speech recognition relies heavily on a well-trained domain-matched language model. Mining user data can only yield a comparably small amount of training data. Adapting a language model with it would barely yield a robust language model for newly learned words, and their usage style may differ in conversational speech.

For the above reasons, we decided not to attempt to adapt vocabulary and language model. Instead, we use a fixed broad-domain vocabulary and language model for large-vocabulary recognition, and augment this system with maintenance-free *phonetic search* to cover new words and mismatched domains.

4.1.2 Phonetic Recognition

The simplest phonetic recognizer is a regular recognizer with the vocabulary replaced by the list of phonemes of the language, and the language model replaced by a phoneme M-gram. However, such phonetic language model is much weaker than a word language model. This results in poor accuracy and inefficient search.

Instead, our recognizer uses "phonetic word fragments" (groups of phonemes similar to syllables or half-syllables) as its vocabulary and in the language model. This provides phonotactic constraints for efficient decoding and accurate phoneme-boundary decisions, while remaining independent of any specific vocabulary. A set of about 600 fragments was automatically derived from the language-model training set by a bottom-up grouping procedure (Klakow, 1998; Ng, 2000; Seide and Yu, 2004). Example fragments are /-k-ih-ng/ (the syllable -*king*), /ih-n-t-ax-r-/ (*inter-*), and /ih-z/ (the word *is*).

With this, lattices are generated using the common Viterbi decoder with word-pair approximation (Schwartz *et al.*, 1994; Ortmanns *et al.*, 1996). The decoder has been modified to keep track of individual phoneme boundaries and scores. These are recorded in the lattices, while fragment-boundary information is discarded. This way, phoneme lattices are generated.

In the results section we will see that, even with a well-trained domain-matching word-level language model, searching phoneme lattices can yield search accuracies comparable with word-level search, and that the best performance is achieved by combining both into a hybrid word/phoneme system.

4.2 Unified Hybrid Lattice Representation

Combining word and phonetic search is desirable because they are complementary: Word-based search yields better precision, but has a recall issue for unknown and rare words, while phonetic search has very good recall but suffers from poor precision especially for short words.

Combining the two is not trivial. Several strategies are discussed in (Yu and Seide, 2004), including using a hybrid recognizer, combining lattices from two separate recognizers, and combining the results of two separate systems. Both hybrid recognizer configuration and lattice combination turned out difficult because of the different dynamic range of scores in word and phonetic paths.

We found it beneficial to convert both lattices into posterior-based representations called *posterior lattices* first, which are then merged into a hybrid posterior lattice. Search is performed in a hybrid lattice in a unified manner using both phonetic and word representations as "alternative pronunciation" of the query, and summing up the resulting phrase posteriors.

Posterior lattices are like regular lattices, except that they do not store acoustic likelihoods, language model probabilities, and precomputed forward/backward probabilities, but *arc and node posteriors*. An arc's posterior is the probability that the arc (with its associated word or phoneme hypothesis) lies on the correct path, while a node posterior is the probability that the correct path connects two word/phoneme hypotheses through this node. In our actual system, a node is only associated with a point in time, and the node posterior is the probability of having a word or phoneme boundary at its associated time point.

The inclusion of node posteriors, which to our knowledge is a novel contribution of this paper, makes an exact computation of phrase posteriors from posterior lattices possible. In the following we will explain this in detail.

4.2.1 Arc and Node Posteriors

A lattice $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{start}}, n_{\text{end}})$ is a directed acyclic graph (DAG) with $\mathcal{N}$ being the set of nodes, $\mathcal{A}$ is the set of arcs, and n_{start}, $n_{\text{end}} \in \mathcal{N}$ being the unique initial and unique final node, respectively. Nodes represent times and possibly context conditions, while arcs represent word or phoneme hypotheses.[3]

Each node $n \in \mathcal{N}$ has an associated time $t[n]$ and possibly an acoustic or language-model context condition. Arcs are 4-tuples $a = (S[a], E[a], I[a], w[a])$. $S[a]$, $E[a]$

[3]Alternative definitions of lattices are possible, e.g. nodes representing words and arcs representing word transitions.

$\in \mathcal{N}$ denote the start and end node of the arc. $I[a]$ is the arc label[4], which is either a word (in word lattices) or a phoneme (in phonetic lattices). Last, $w[a]$ shall be a weight assigned to the arc by the recognizer. Specifically, $w[a] = p_{\text{ac}}(a)^{1/\lambda} \cdot P_{\text{LM}}(a)$ with acoustic likelihood $p_{\text{ac}}(a)$, language model probability P_{LM}, and language-model weight λ.

In addition, we define *paths* $\pi = (a_1, \cdots, a_K)$ as *sequences* of connected arcs. We use the symbols S, E, I, and w for paths as well to represent the respective properties for entire paths, i.e. the path start node $S[\pi] = S[a_1]$, path end node $E[\pi] = E[a_K]$, path label sequence $I[\pi] = (I[a_1], \cdots, I[a_K])$, and total path weight $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

Finally, we define $\Pi(n_1, n_2)$ as the entirety of all paths that start at node n_1 and end in node n_2: $\Pi(n_1, n_2) = \{\pi | S[\pi] = n_1 \wedge E[\pi] = n_2\}$.

With this, the phrase posteriors defined in Eq. 2 can be written as follows.

In the simplest case, Q is a single word token. Then, the phrase posterior is just the word posterior and, as shown in e.g. (Wessel *et al.*, 2000) or (Evermann *et al.*, 2000), can be computed as

$$\begin{aligned} P(*, t_s, Q, t_e, *|O) &= \frac{\sum\limits_{\substack{\pi=(a_1,\cdots,a_K) \in \Pi(n_{\text{start}}, n_{\text{end}}): \\ \exists l: [S[a_l]] = t_s \wedge t[E[a_l]] = t_e \wedge I[a_l] = Q}} w[\pi]}{\sum\limits_{\pi \in \Pi(n_{\text{start}}, n_{\text{end}})} w[\pi]} \\ &= \sum_{\substack{a \in \mathcal{A}: t[S[a]] = t_s \\ \wedge t[E[a]] = t_e \wedge I[a] = Q}} P_{\text{arc}}[a] \qquad (3) \end{aligned}$$

with $P_{\text{arc}}[a]$ being the *arc posterior* defined as

$$P_{\text{arc}}[a] = \frac{\alpha_{S[a]} \cdot w[a] \cdot \beta_{E[a]}}{\alpha_{n_{\text{end}}}}$$

with the *forward/backward probabilities* α_n and β_n defined as:

$$\begin{aligned} \alpha_n &= \sum_{\pi \in \Pi(n_{\text{start}}, n)} w[\pi] \\ \beta_n &= \sum_{\pi \in \Pi(n, n_{\text{end}})} w[\pi]. \end{aligned}$$

α_n and β_n can conveniently be computed from the word lattices by the well-known forward/backward recursion:

$$\begin{aligned} \alpha_n &= \begin{cases} 1.0 & n = n_{\text{start}} \\ \sum\limits_{a: E[a] = n} \alpha_{S[a]} \cdot w[a] & \text{otherwise} \end{cases} \\ \beta_n &= \begin{cases} 1.0 & n = n_{\text{end}} \\ \sum\limits_{a: S[a] = n} w[a] \cdot \beta_{E[a]} & \text{otherwise.} \end{cases} \end{aligned}$$

[4]Lattices are often interpreted as weighted finite-state acceptors, where the arc labels are the *input symbols*, hence the symbol I.

Now, in the general case of multi-label queries, the phrase posterior can be computed as

$$\begin{aligned} &P(*, t_s, Q, t_e, *|O) \\ &= \sum_{\substack{\pi=(a_1,\cdots,a_K): \\ t[S[\pi]] = t_s \wedge t[E[\pi]] = t_e \wedge I[\pi] = Q}} \frac{P_{\text{arc}}[a_1] \cdots P_{\text{arc}}[a_K]}{P_{\text{node}}[S[a_2]] \cdots P_{\text{node}}[S[a_K]]} \end{aligned}$$

with $P_{\text{node}}[n]$, the *node posterior*[5], defined as

$$P_{\text{node}}[n] = \frac{\alpha_n \cdot \beta_n}{\alpha_{n_{\text{end}}}}. \qquad (4)$$

4.2.2 Advantages of Posterior Lattices

The posterior-lattice representation has several advantages over traditional lattices. First, lattice storage is reduced because only one value (node posterior) needs to be stored per node instead of two (α, β)[6]. Second, node and arc posteriors have a smaller and similar dynamic range than α_n, β_n, and $w[a]$, which is beneficial when the values should be stored with a small number of bits.

Further, for the case of word-based search, the summation in Eq. 3 can also be precomputed by merging all lattice nodes that carry the same time label, and merging the corresponding arcs by summing up their arc posteriors. In such a "pinched" lattice, word posteriors for single-label queries can now be looked up directly. However, posteriors for multi-label strings cannot be computed precisely anymore. Our experiments have shown that the impact on ranking accuracy caused by this approximation is neglectable. Unfortunately, we have also found that the same is not true for phonetic search.

The most important advantage of posterior lattices for our system is that they provide a way of combining the word and phoneme lattices into a single structure – by simply merging their start nodes and their end nodes. This allows to implement hybrid queries in a single unified search, treating the phonetic and the word-based representation of the query as alternative pronunciations.

5 Lattice Indexing

Searching lattices is time-consuming. It is not feasible to search large amounts of audio. To deal with hundreds or even thousands of hours of audio, we need some form of inverted indexing mechanism.

This is comparably straight-forward when indexing text. It is also not difficult for indexing word lattices. In both case, the set of words to index is known. However, indexing phoneme lattices is very different, because theoretically any phoneme string could be an indexing item.

[5]Again, mind that in our lattice formulation word/phoneme hypotheses are represented by arcs, while nodes just represent connection points. The node posterior is the probability that the correct path passes through a connection point.

[6]Note, however, that storage for the traditional lattice can also be reduced to a single number per node by weight pushing (Saraclar and Sproat, 2004), using an algorithm that is very similar to the forward/backward procedure.

We address this by our M-gram lattice-indexing scheme. It was originally designed for phoneme lattices, but can be – and is actually – used in our system for indexing word lattices.

First, audio files are clipped into homogeneous segments. For an audio segment i, we define the *expected term frequency* (ETF) of a query string Q as summation of phrase posteriors of all hits in this segment:

$$\begin{aligned}\mathrm{ETF}_i(Q) &= \sum_{\forall t_s, t_e} P(*, t_s, Q, t_e, * | O^i) \\ &= \sum_{\pi \in \Pi^i : I[\pi] = Q} p[\pi]\end{aligned}$$

with Π^i being the set of all paths of segment i.

At indexing time, ETFs of a list of M-grams for each segment are calculated. They are stored in an inverted structure that allows retrieval by M-gram.

In search time, the ETFs of the query string are estimated by the so-called "M-gram approximation". In order to explain this concept, we need to first introduce $P(Q|O^i)$ – the probability of observing query string Q at any word boundary in the recording O^i. $P(Q|O^i)$ has a relationship with ETF as

$$\mathrm{ETF}_i(Q) = \tilde{N}_i \cdot P(Q|O^i)$$

with $\tilde{N}_i$ being the expected number of words in the segment i. It can also be computed as

$$\tilde{N}_i = \sum_{n \in \mathcal{N}_i} p[n],$$

where $\mathcal{N}_i$ is the node set for segment i.

Like the M-gram approximation in language-model theory, we approximate $P(Q|O^i)$ as

$$\begin{aligned}P(Q|O^i) &\approx \tilde{P}(Q|O^i) \\ &= \prod_{k=1}^{l} \tilde{P}(q_k | q_{k-M+1}, \cdots, q_{k-1}, O^i),\end{aligned}$$

while the right-hand items can be calculated from M-gram ETFs:

$$\begin{aligned}&\tilde{P}(q_k | q_{k-M+1}, \cdots, q_{k-1}, O^i) \\ &\quad = \frac{\mathrm{ETF}_i(q_{k-M+1}, \cdots, q_k)}{\mathrm{ETF}_i(q_{k-M+1}, \cdots, q_{k-1})}.\end{aligned}$$

The actual implementation uses only M-grams extracted from a large background dictionary, with a simple backoff strategy for unseen M-grams, see (Yu *et al.*, 2005) for details.

The resulting index is used in a two stage-search manner: The index itself is only used as the first stage to determine a short-list of promising segments that may contain the query. The second stage involves a linear lattice search to get final results.

Table 1: Test corpus summary.

test set	duration	#segments	keyword set (incl. OOV)
ICSI meetings	2.0h	429	1878 (96)
SWBD eval2000	3.6h	742	2420 (215)
SWBD rt03s	6.3h	1298	2325 (236)
interviews (phone)	1.1h	267	1057 (49)
interviews (lapel)	1.0h	244	1629 (107)

6 Results

6.1 Setup

We have evaluated our system on five different corpora of recorded conversations:

- one meeting corpus (NIST "RT04S" development data set, ICSI portion, (NIST, 2000-2004))
- two eval sets from the switchboard (SWBD) data collection ("eval 2000" and "RT03S", (NIST, 2000-2004))
- two in-house sets of interview recordings of about one hour each, one recorded over the telephone, and one using a single microphone mounted in the interviewee's lapel.

For each data set, a keyword list was selected by an automatic procedure (Seide and Yu, 2004). Words and multi-word phrases were selected from the reference transcriptions if they occurred in at most two segments. Example keywords are *overseas*, *olympics*, and *"automated accounting system"*. For the purpose of evaluation, those data sets are cut into segments of about 15 seconds each. The size of the corpora, their number of segments, and the size of the selected keyword set are given in Table 1.

The acoustic model we used is trained on 309h of the Switchboard corpus (SWBD-1). The LVCSR language model was trained on the transcriptions of the Switchboard training set, the ICSI-meeting training set, and the LDC Broadcast News 96 and 97 training sets. No dedicated training data was available for the in-house interview recordings. The recognition dictionary has 51388 words. The phonetic language model was trained on the phonetic version of the transcriptions of SWBD-1 and Broadcast News 96 plus about 87000 background dictionary entries, a total of 11.8 million phoneme tokens.

To measure the search accuracy, we use the "Figure Of Merit" (FOM) metric defined by NIST for word-spotting evaluations. In its original form, it is the average of detection/false-alarm curve taken over the range [0..10] false alarms per hour per keyword. Because manual word-level alignments of our test sets were not available, we modified the FOM such that a correct hit is a 15-second segment that contains the key phrase.

Besides FOM, we use a second metric – "Top Hit Precision" (THP), defined as the correct rate of the best ranked hit. If no hit is returned for an existing query term, it is counted as an error. Both of these metrics are relevant measures in our known-item search.

Table 2: Baseline transcription word-error rates (WER) as well as precision (P), recall (R), FOM and THP for searching the transcript.

test set	WER [%]	P [%]	R [%]	FOM [%]	THP [%]
ICSI meetings	44.1	80.6	43.8	43.6	43.6
SWBD eval2000	39.0	79.6	41.1	41.1	41.1
SWBD rt03s	45.2	72.6	36.3	36.3	36.0
interviews (phone)	57.7	68.8	31.6	29.3	31.3
interviews (lapel)	62.8	80.1	32.0	30.2	32.1
average	49.8	76.3	37.0	36.1	36.8

Table 3: Comparison of search accuracy for word, phoneme, and hybrid lattices.

test set	word	phoneme	hybrid
Figure Of Merit (FOM) [%]			
ICSI meetings	72.1	81.2	88.2
SWBD eval2000	71.3	80.4	87.3
SWBD rt03s	66.4	76.9	84.2
interviews (phone)	60.6	73.7	83.3
interviews (lapel)	59.0	70.2	77.7
average	65.9	76.5	84.1
INV words only	69.4	77.0	84.7
OOV words only	0	73.8	73.8
Top Hit Precision (THP) [%]			
ICSI meetings	67.2	65.0	78.7
SWBD eval2000	67.1	63.6	77.9
SWBD rt03s	59.6	59.1	71.7
interviews (phone)	55.7	64.4	73.1
interviews (lapel)	55.6	59.7	71.2
average	61.0	62.4	74.5
INV words only	64.5	62.4	75.3
OOV words only	0	60.5	60.5

6.2 Word/Phoneme Hybrid Search

Table 2 gives the LVSCR transcription word-error rates for each set. Almost all sets have a word-error rates above 40%. Searching those speech recognition transcriptions results in FOM and THP values below 40%.

Table 3 gives results of searching in word, phoneme, and hybrid lattices. First, for all test sets, word-lattice search is drastically better than transcription-only search.

Second, comparing word-lattice and phoneme-lattice search, phoneme lattices outperforms word lattices on all tests in terms of FOM. This is because phoneme lattice has better recall rate. For THP, word lattice search is slightly better except on the interview sets for which the language model is not well matched. Hybrid search leads to a substantial improvement over each (27.6% average FOM improvement and 16.2% average THP improvement over word lattice search). This demonstrates the complementary nature of word and phoneme search.

We also show results separately for known words (in-vocabulary, INV) and out-of-vocabulary words (OOV). Interestingly, even for known words, hybrid search leads to a significant improvement (get 22.0% for FOM and 16.7% for THP) compared to using word lattices only.

6.3 Effect of Node Posterior

In Section 4.2, we have shown that phrase posteriors can be computed from posterior lattices if they include both arc and node posteriors (Eq. 4). However, posterior representations of lattices found in literature only include word (arc) posteriors, and some posterior-based systems simply ignore the node-posterior term, e.g. (Chelba and Acero, 2005). In Table 4, we evaluate the impact on accuracy when this term is ignored. (In this experiment, we bypassed the index-lookup step, thus the numbers are slightly different from Table 3.)

We found that for word-level search, the effect of node posterior compensation is indeed neglectable. However, for phonetic search it is not: We observe a 4% relative FOM loss.

6.4 Index Lookup and Linear Search

Section 5 introduced a two-stage search approach using an M-gram based indexing scheme. How much accuracy is lost from incorrectly eliminating correct hits in the first (index-based) stage? Table 5 compares three setups. The first column shows results for linear search only: no index lookup used at all, a complete linear search is performed on all lattices. This search is optimal but does not scale up to large database. The second column shows index lookup *only*. Segments are ranked by the approximate M-gram based ETF score obtained from the index. The third column shows the two-stage results.

The index-based two-stage search is indeed very close to a full linear search (average FOM loss of 1.2% and THP loss of 0.2% points). A two-stage search takes under two seconds and is mostly independent of the database size. In other work, we have applied this technique successfully to search a database of nearly 200 hours.

6.5 The System

Fig. 2 shows a screenshot of a research prototype for a search-enabled audio notebook. In addition to a note-taking area (bottom) and recording controls, it includes a rich audio browser showing speaker segmentation and automatically identified speaker labels (both not scope of this paper). Results of keyword searches are shown as color highlights, which are clickable to start playback at that position.

7 Conclusion

In this paper, we have presented a system for searching recordings of conversational speech, particularly meet-

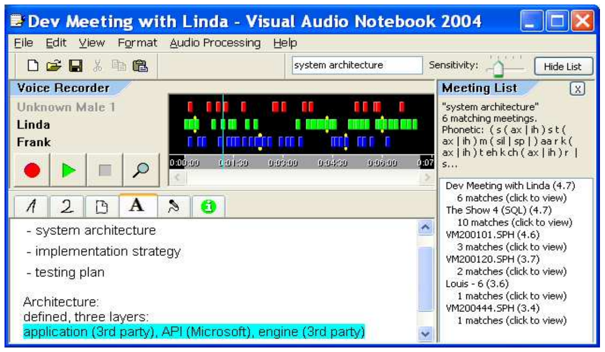
Table 4: Effect of ignoring the node-posterior term in phrase-posterior computation (shown for ICSI meeting set only).

FOM	word	phoneme
exact computation	72.1	82.3
node posterior ignored	72.0	79.2
relative change [%]	-0.1	-3.8

Table 5: Comparing the effect of lattice indexing. Shown is unindexed "linear search," index lookup only (segments selected via the index without subsequent linear search), and the combination of both.

test set	linear search	index lookup	two-stage
Figure Of Merit (FOM) [%]			
ICSI meetings	88.6	86.4	88.2
SWBD eval2000	88.7	86.5	87.3
SWBD rt03s	87.3	85.1	84.2
interviews (phone)	83.8	81.2	83.3
interviews (lapel)	78.3	76.1	77.7
average	85.3	83.1	84.1
Top Hit Precision (THP) [%]			
ICSI meetings	78.8	70.7	78.7
SWBD eval2000	78.0	71.4	77.9
SWBD rt03s	71.9	65.7	71.7
interviews (phone)	73.8	64.6	73.1
interviews (lapel)	70.8	65.9	71.2
average	74.7	67.7	74.5

ings and telephone conversations. We identified user requirements as accurate ranking of phrase matches, domain independence, and reasonable response time. We have addressed these by hybrid word/phoneme lattice search and a supporting indexing scheme. Unlike many other spoken-document retrieval systems, we search recognition alternates instead of only speech recognition transcripts. This yields a significant improvement of keyword spotting accuracy. We have combined word-level search with phonetic search, which not only enables the system to handle the open-vocabulary problem, but also substantially improves in-vocabulary accuracy. We have proposed a posterior-lattice representation that allows for unified word and phoneme indexing and search. To speed up the search process, we proposed M-gram based lattice indexing, which extends our open vocabulary search ability for large collection of audio. Tested on five different recording sets including meetings, conversations, and interviews, a search accuracy (FOM) of 84% has been achieved – dramatically better than searching speech recognition transcripts (under 40%).



system architecture

- system architecture
- implementation strategy
- testing plan

Architecture:
defined, three layers:
application (3rd party), API (Microsoft), engine (3rd party)

"system architecture"
6 matching meetings.
Phonetic: (s (ax | ih) s t (ax | ih) m (sil | sp |) aa r k (ax | ih) t eh k ch (ax | ih) r | s...

Dev Meeting with Linda (4.7)
6 matches (click to view)
The Show 4 (SQL) (4.7)
10 matches (click to view)
VM200101.SPH (4.6)
3 matches (click to view)
VM200120.SPH (3.7)
2 matches (click to view)
Louis - 6 (3.6)
1 matches (click to view)
VM200444.SPH (3.4)
1 matches (click to view)

Figure 2: Screenshot of our research prototype of a search-enabled audio notebook.

8 Acknowledgements

The authors wish to thank our colleagues Asela Gunawardana, Patrick Nguyen, Yu Shi, and Ye Tian for sharing their Switchboard setup and models with us; and Frank Soong for his valuable feedback on this paper.

References

C. Allauzen, M. Mohri, M. Saraclar, General indexation of weighted automata – application to spoken utterance retrieval. *Proc. HLT'2004*, Boston, 2004.

C. Chelba and A. Acero, Position specific posterior lattices for indexing speech. *Proc. ACL'2005*, Ann Arbor, 2005.

Mark Clements *et al.*, Phonetic Searching vs. LVCSR: How to find what you really want in audio archives. *Proc. AVIOS'2001*, San Jose, 2001.

G. Evermann *et al.*, Large vocabulary decoding and confidence estimation using word posterior probabilities. *Proc. ICASSP'2000*, Istanbul, 2000

J. Garofolo, TREC-9 Spoken Document Retrieval Track. National Institute of Standards and Technology, `http://trec.nist.gov/pubs/trec9/sdrt9_slides/sld001.htm`.

D. A. James and S. J. Young, A fast lattice-based approach to vocabulary-independent wordspotting. *Proc. ICASSP'1994*, Adelaide, 1994.

D. Klakow. Language-model optimization by mapping of corpora. *Proc. ICASSP'1998*.

Beth Logan *et al.*, An experimental study of an audio indexing system for the web. *Proc. ICSLP'2000*, Beijing, 2000.

Beth Logan *et al.*, Word and subword indexing approaches for reducing the effects of OOV queries on spoken audio. *Proc. HLT'2002*, San Diego, 2002.

Kenney Ng, Subword-based approaches for spoken document retrieval. PhD thesis, Massachusetts Institute of Technology, 2000.

NIST Spoken Language Technology Evaluations, `http://www.nist.gov/speech/tests/`.

S. Ortmanns, H. Ney, F. Seide, and I. Lindam, A comparison of the time conditioned and word conditioned search techniques for large-vocabulary speech recognition. *Proc. ICSLP'1996*, Philadelphia, 1996.

S. E. Robertson, The probability ranking principle in IR. Journal of Documentation 33 (1977).

M. Saraclar, R. Sproat, Lattice-based search for spoken utterance retrieval. *Proc. HLT'2004*, Boston, 2004.

P. Schäuble *et al.*, First experiences with a system for content based retrieval of information from speech recordings. *Proc. IJCAI'1995*, Montreal, 1995.

R. Schwartz *et al.*, A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. *Proc. ICSLP'1994*, Yokohama, 1994.

F. Seide, P. Yu, *et al.*, Vocabulary-independent search in spontaneous speech. *Proc. ICASSP'2004*, Montreal, 2004.

Lisa Joy Stifelman, The Audio Notebook. PhD thesis, Massachusetts Institute of Technology, 1997.

F. Wessel, R. Schlüter, and H. Ney, Using posterior word probabilities for improved speech recognition. *Proc. ICASSP'2000*, Istanbul, 2000.

P. Yu, F. Seide, A hybrid word / phoneme-based approach for improved vocabulary-independent search in spontaneous speech. *Proc. ICLSP'04*, Jeju, 2004.

P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-Independent Indexing of Spontaneous Speech, to appear in IEEE transaction on Speech and Audio Processing, Special Issue on Data Mining of Speech, Audio and Dialog.

Learning a Spelling Error Model from Search Query Logs

Farooq Ahmad
Department of Electrical and
Computer Engineering
University of Alberta
Edmonton, Canada
farooq@ualberta.ca

Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, Canada
kondrak@cs.ualberta.ca

Abstract

Applying the noisy channel model to search query spelling correction requires an error model and a language model. Typically, the error model relies on a weighted string edit distance measure. The weights can be learned from pairs of misspelled words and their corrections. This paper investigates using the Expectation Maximization algorithm to learn edit distance weights directly from search query logs, without relying on a corpus of paired words.

1 Introduction

There are several sources of error in written language. Typing errors can be divided into two groups (Kucich, 1992): typographic errors and cognitive errors. Typographic errors are the result of mistyped keys and can be described in terms of keyboard key proximity. Cognitive errors on the other hand, are caused by a misunderstanding of the correct spelling of a word. They include phonetic errors, in which similar sounding letter sequences are substituted for the correct sequence; and homonym errors, in which a word is substituted for another word with the same pronunciation but a different meaning. Spelling errors can also be grouped into errors that result in another valid word, such as homonym errors, versus those errors that result in a non-word. Generally non-word errors are easier to detect and correct. In addition to its traditional use in word processing, spelling correction also has applications in optical character recognition and handwriting recognition. Spelling errors in this context are caused by inaccurate character recognition.

Spelling correction is a well developed research problem in the field of computational linguistics. The first dictionary based approach to spelling correction (Damerau, 1964) considers all words that can not be found in a dictionary as misspellings. The correct word is found by making a single edit operation (insertion, deletion, or substitution) on the misspelled word and re-checking the dictionary for the inclusion of the altered version. This method works well for correcting most typos, but often misspelled words are off by more than one character. A method of quantifying string-to-string distance is introduced in (Wagner and Fischer, 1974), allowing the consideration of multiple edit operations when determining candidate corrections. Each edit operation is assigned a fixed cost. Edit operations, though, can be more accurately modelled by considering every possible insertion, deletion, and substitution operation individually instead of having a fixed cost for each operation. For example, the application of probabilistic models to spelling correction is explored in (Kernighan, Church, and Gale, 1990), in which a confusion matrix describes the probability of each letter being substituted for another. The Bayesian noisy channel model is used to determine the the error probabilities, with the simplifying assumption that each word has at most one spelling error. In (Ristad and Yianilos, 1997), a probabilistic model of edit distance is learned from pairs of misspelled words and their corrections. This extends Kernighan's approach by allowing multiple edit operations rather than assuming a single edit. The probability of edit operations is learned from a corpus of pairs of misspelled words and corrections.

Search query correction is an interesting branch of spelling correction. Due to the wide variety of search queries, dictionary based spelling correction is not adequate for correcting search terms. The concept of using query logs to aid in spelling correction is explored in (Brill and Cucerzan, 2004). It is noted that using traditional Levenshtein distance as an error model can lead to inappropriate corrections, so a weighted distance measure is used instead.

This paper focuses on deriving a language model and probabilistic error model directly from search query logs without requiring a corpus of misspelled words paired with their corrections. The task of search query spelling correction is analyzed, and an implementation of the Expectation Maximization (EM) algorithm to learn an error model is described, with reference to similar approaches. In Section 2, the make-up of search queries is analyzed in the context of spelling correction. Section 3 details the noisy channel model spelling correction framework and describes how the EM algorithm is applied to learn an error model. The learned error model is explored in Section 4. The derived model is tested in Section 5 by comparing its performance in the single word spelling correction task to popular spell checking applications. Finally, conclusions and directions for future work are presented in Section 6.

2 Analysis of Search Queries

Search queries present a difficult challenge for traditional spelling correction algorithms. As mentioned above, dictionary-based approaches cannot be used since many search terms include words and names that are not well established in the language. Furthermore, search queries typically consist of a few key words rather than grammatically correct sentences, making grammar-based approaches inappropriate. In addition, spelling errors are more common in search queries than in regular written text, as approximately 10-15 % of search queries contain a misspelling (Brill and Cucerzan, 2004). The suitability of query logs as a corpus for spelling correction is investigated in this section.

The metaspy website[1] displays search queries submitted to the popular metacrawler search engine in real time. Over a period of five days in the last week of March 2005, 580,000 queries were extracted from the site. Several interesting observations can be made from the analysis of the search queries.

[1]www.metaspy.com

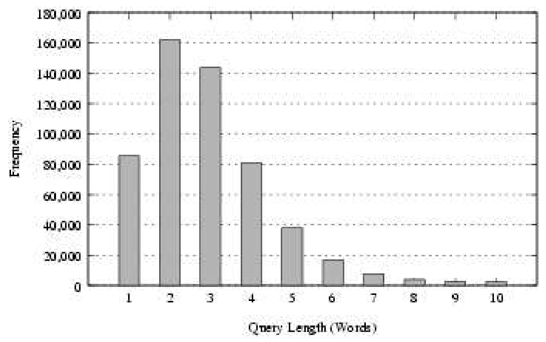


Figure 1: Query Length Frequency Histogram

2.1 Query Length

On average, each query consisted of approximately 3 words. Figure 1 shows the distribution of query lengths.

As illustrated in Figure 1, over 80% of queries include more than one search term. Thus word n-gram probabilities provide useful statistical knowledge that can be exploited to improve spelling correction. Although word cooccurrences are not used for spelling correction in this paper, the possibilities for n-gram analysis are explored in Section 3.2. The longer queries (>5 terms) often contain quotations, song lyric excerpts or very specific product names.

The frequency of words in written text has been shown to follow Zipf's law. That is, if the words are ordered in terms of frequency, the relationship between frequency and rank can be approximated with the following equation.

$$F \approx \frac{C}{r^m} \tag{1}$$

where F is the frequency, r is rank, C is a constant, and m is an exponent close to 1. In logarithmic form,

$$\log(F) = \log(C) - m * \log(r) \tag{2}$$

The frequency and rank of search query tokens approximately follow the same distribution, with some deviation at the high and low ends. Figure 2 shows the frequency distribution for dictionary and

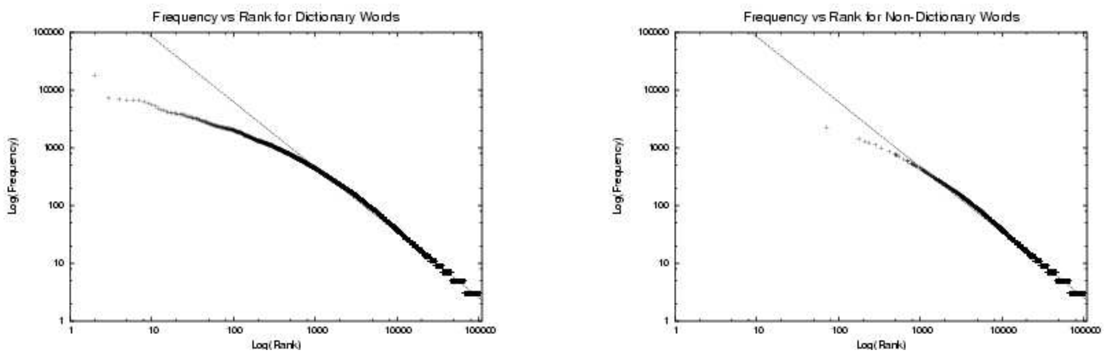


Figure 2: Token Frequency vs. Rank for Dictionary and Non-Dictionary Words

non-dictionary search query tokens. The word list available on most Unix systems */usr/dict/words* is a comprehensive list that contains 96,274 words, including names, plurals, verbs in several tenses, and colloquialisms. Following tokenization of the query logs, the tokens were divided into dictionary and non-dictionary words. The frequency-rank relationship is similar for both types of words, except that nearly all of the 100 most frequent query tokens are dictionary words. The exponent m, the (negative) slope of the linear best fit lines shown in Figure 2, was determined to be 1.11 for dictionary words, and 1.14 for non-dictionary words. As in (Baeza-Yates, 2005), the exponent is slightly higher than 1, partly due to the less frequent use of function words such as *the* in search queries relative to formal writing.

Although the majority of search tokens can be found in a standard dictionary, a large proportion of the less common queries are not dictionary words. In fact, 73% of unique word types were not found in the dictionary. Taking token frequency into consideration, these non-dictionary tokens account for approximately 20% of query search words, including correctly and incorrectly spelled words. However, the majority of the non-dictionary tokens are correctly spelled words, illustrating the unsuitability of traditional dictionary based spelling correction for search query correction.

What are these non-dictionary words? An analysis of the top two hundred non-dictionary words in the query logs allows categorization into a few main groups. The percentage of non-dictionary words belonging to each category, and some examples from each category are shown in Table 1. The first category, *e-speak*, includes words and abbreviations that are commonly used online, but have not crossed over into common language. This category includes words such as *pics, multiplayer*, and *clipart*. The second category is closely related to the first, and includes company and product names, such as *google, xbox*, and *hotmail*. Many of these terms refer to online entities or computer games. Incorrectly spelled words are another main class of non-dictionary tokens. Among the top 20 non-dictionary tokens are words with missing punctuation, such as *womens* and *childrens*, or with missing spaces, such as *realestate*. Names of people and locations are also common search queries, as well as medical terminology. Finally, foreign words make up another class of words that are not found in an (English) dictionary. The 20 highest frequency non-dictionary tokens from the extracted query logs are *pics, html, multiplayer, googletestad, google, xbox, childrens, ebay, angeles, hotmail, womens, ERBS, clipart, playstation, ratzinger, Ilios, lettre, realestate, tech* and *mallorca*.

	Word Class	**Percent**	**Examples**
1	E-speak & new words	45%	pics, html, multiplayer, clipart, mpeg, midi
2	Companies & Products	18%	google, xbox, ebay, hotmail , playstation
3	Proper Names	16%	(los) angeles, ratzinger, ilios, mallorca
4	Medical terms	5%	ERBS, mesothelioma, neuropsychological, alzheimers
5	Misspellings	9%	womens, realestate
6	Foreign Words	6%	lettre, para

Table 1: Classes of Non-Dictionary Words

3 Spelling Correction for Search Queries

The spelling correction problem can be considered in terms of the noisy channel model, which considers the misspelled word v to be a corrupted version of the correctly spelled word w.

$$P(w|v) = \frac{P(v|w)P(w)}{P(v)} \tag{3}$$

Finding the best candidate correction W involves maximizing the above probability.

$$W = argmax_w P(v|w)P(w) \tag{4}$$

The denominator $P(v)$ in Equation 3 is the same for all w and can be eliminated from the calculation. $P(v|w)$ models the errors that corrupt string w into string v, and $P(w)$ is the language model, or prior probability, of word w.

3.1 Error Model

Given two strings v and w, $P(v|w)$ is the probability that v is transmitted given that the desired word is w. One method of describing the noise model is to consider $P(v|w)$ to be proportional to the number of edit operations required to transform w into v. This gives

$$P(v|w) \propto ED(v, w) \tag{5}$$

where $ED(v, w)$ is the edit distance between v and w.

The traditional edit distance calculation assigns a fixed cost for each insertion, deletion, and substitution operation. For example, each insertion and deletion may be assigned a cost of 1, while substitutions are assigned a cost of 1.5. The edit distance calculation can be accomplished by dynamic programming.

The error model can be improved if each edit operation is considered separately, rather than assigning a fixed cost to each operation. For example, the substitution of the letter i for the letter e may be much more likely than k for e. Thus if a string S_1 differs from string S_2 by one $e \rightarrow i$ substitution, it should be considered more similar to S_2 than a string S_3 that differs from S_1 by an $e \rightarrow k$ substitution.

Generating an accurate error model that considers each edit operation individually requires learning edit distance weights. As described in (Ristad and Yianilos, 1997), character-to-character edit distance costs $ED(e)$ can be related to edit probability $P(e)$ by means of the equation:

$$ED(e) = -\log[P(e)] \tag{6}$$

where e is an edit operation consisting of a substitution of one alphanumeric character for another ($c_1 \rightarrow c_2$), an insertion ($\_ \rightarrow c_1$), or a deletion ($c_1 \rightarrow \_$).

Thus higher probability edits will have lower edit distances, and the string to string edit distance calculation proceeds in the same way as the traditional calculation. This convenient representation allows whole string-to-string edit probability to be expressed in terms of the edit distance of the edit sequence $[e_1 ... e_n]$:

$$\begin{aligned} P(w|v) &= \Pi P(e_i) \\ &= P(e_1) * P(e_2) * ... * P(e_n) \end{aligned} \tag{7}$$

Taking the log of both sides gives

$$\begin{aligned} \log[P(w|v)] =& \log[P(e_1)] + \log[P(e_2)] \\ & + ... + \log[P(e_n)] \end{aligned} \tag{8}$$

Finally, by combining 6 and 8 we can relate the probability of misspelling a string w as v to string-to-string edit distance.

$$\log[P(w|v)] = -ED(w, v) \tag{9}$$

The edit probabilities can be estimated using the expectation maximization (EM) algorithm as described in Section 3.3.

3.2 Language Model

Along with the error model, a language model is used to determine the most likely correction for every input query. Often, spelling correction programs use N-gram language models that use nearby words to help determine the most probable correction. For example, it is noted in (Brill and Cucerzan, 2004) that employing a trigram language model can substantially improve performance relative to a unigram model. However, if search query logs are not very large, bigram or trigram data may be too sparse to be helpful. Nevertheless, a word unigram model can be used for training the error model. The unigram model is determined by tokenizing the query logs and determining the frequency of each token. The language model $P(w)$ is the frequency of the word $C(w)$ divided by the total number of tokens N in the query log:

$$P(w) = \frac{C(w)}{N} \tag{10}$$

Add-One smoothing is used to account for words not present in query logs.

3.3 Determining Edit Probabilities with Expectation Maximization

The EM algorithm is used to determine the parameters of the probability distribution for a given a set of data. It can be considered to be a soft-clustering algorithm: given several data points, the task is to find the cluster parameters which best represent the data. The EM algorithm is applied iteratively to each data point in a two-step process; the expectation step determines the degree to which data agrees with each cluster/hypothesis, and the maximization step updates the parameters to reflect the inclusion of the new data.

Prior to running the EM algorithm, the edit distance table is seeded with initial values. The initialization stage assigns high probability (low edit distance) to characters being typed correctly, and a lower probability for character substitutions. For each character l, substitution distance is equally distributed over all other characters and the deletion operation ($l \rightarrow \_$). Specifically the initial probability for a character match was set to 90%, and the remaining 10% was equally distributed over the other 26 possible substitutions. Essentially, the first edit distance calculated in the EM algorithm will be equivalent to the fixed-weight Levenshtein distance. After this preprocessing stage, the edit probability matrix is iteratively improved with the E-Step and M-Step described below. The operation of the EM algorithm is illustrated in Figure 3.

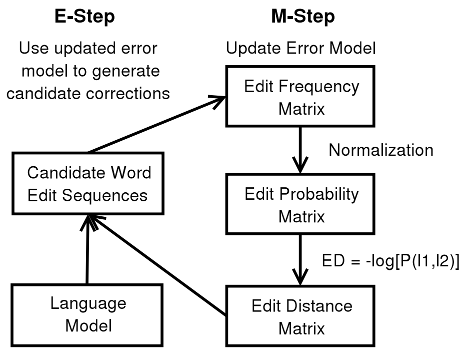


Figure 3: The EM process

For each query token, possible corrections are harvested from the query word list. The entire word list is searched, and any word within a threshold edit distance is considered as a candidate. Since the query logs can be quite large, determining the exact weighted edit distance between the input query and each logged query is quite computationally expensive. Instead, the candidate queries are first narrowed down using a fast approximate string matching algorithm (Wu and Manber, 1990) to determine all candidates within k unweighted edit operations. Then, the candidate queries that are within a second tighter threshold T, based on weighted edit distance, are kept.

$$Candidates(v) = \{w_i | ED(w_i, v) < T\}$$

Generally several words in the query logs will meet the above criteria. The threshold T is chosen to ensure the inclusion of all reasonable corrections, while maintaining a manageable computation time. If T were infinite, every query log token would need to be considered, taking too much time. On the other hand, if T is too small, some corrections may not be considered. In practice, K was set to 3 unweighted edits, and T was set as a constant proportion of word length.

The expectation of each candidate correction is the probability that the word w_i was desired given that the query was v:

$$P(w_i|v) = \frac{P(v|w_i)P(w_i)}{P(v)} \tag{11}$$

where $P(v|w)$ and $P(w)$ are determined using the error and language models described in Equations (9) and (10).

If the value of T is set high enough, it can be assumed that the correct word w is within the set of candidates. So, the sum of probabilities over all candidates is normalized to $P(v)$ in accordance with Bayes Rule of Total Probability.

$$P(v) = \Sigma_j P(v|w_j)P(w_j) \tag{12}$$

Correction	**Error Model**	**Language Model**	**Total Probability**	**Normalized**
equipment	0.0014	0.00078	1.1e-6	0.77
equpment	0.64	5.0e-7	3.4e-7	0.23
equipiment	0.0005	5.0e-7	1.0e-9	0.0005

Table 2: Candidate Corrections for *equpment*

This gives us the following formula for the expectation value

$$P(w_i|v) = \frac{P(v|w_i)P(w_i)}{\Sigma_j P(v|w_j)P(w_j)} \qquad (13)$$

The E-step is used to generate candidate corrections for input query tokens. For example, input query *"equpment"* returns the candidate corrections and their probabilities shown in Table 2.

Note that several incorrectly spelled words, including *"equpment"* itself, are given as candidate corrections. However, the language model derived from the query logs assigns a low probability to the incorrect candidates. In the case of a correctly spelled query, the most likely candidate correction is the word itself. However, occasionally there is a correctly spelled but infrequent word within a small edit distance of another more common word. In this case, the language model will bias the correction probability in favor of an incorrect edit. Nevertheless, overall these cases do not seem to cause a significant impact on the error model except in the case of plural nouns as discussed in Section 4.

The maximization step updates the edit distance probabilities and edit distance table to reflect the query considered in the E-Step. For each candidate correction, the required edits are added to the edit frequency table, weighted by the probability of the correction. Then, the probability of an edit for each character is normalized to 1 and the edit probabilities are stored in a table. Finally, Equation 6 is used to generate the edit probability table. For example, for the input query *"equpment"* in response to the first candidate correction ($equpment \rightarrow equipment$), the following substitution frequencies will each be incremented by 0.77: $e \rightarrow e, q \rightarrow q, u \rightarrow u, i \rightarrow \_, p \rightarrow p, m \rightarrow m, e \rightarrow e, n \rightarrow n, t \rightarrow t$. The ($i \rightarrow \_$) edit represents deletion of the letter i.

Letter	**Subs**	**Letter**	**Subs**
a	e_qo	n	_fkb
b	grnw	o	a_ei
c	_ksm	p	nfrm
d	ds_nk	q	glk_
e	ao_i	r	_sdm
f	btpj	s	_mdn
g	o_ks	t	_yir
h	_rab	u	_rio
i	_aue	v	awcm
j	blhm	w	prgk
k	vots	x	gtms
l	r_is	y	ioaje
m	nkvs	z	skmt

Table 3: Most Common Substitutions

4 The Learned Error Model

Approximately 580,000 queries were extracted from the metaspy site over a period of 5 days. After generating a language model by analyzing token frequencies, the EM algorithm was run on a subset of the queries to find the edit probability matrix.

After 15,000 iterations, several patterns can be observed in the edit distance table. The most common edit operations are shown in Table 3. As expected, vowels are most commonly substituted for other vowels. As can be seen in the table, vowel-to-vowel edits are more probable than vowel-to-consonant transitions. The letter e is most commonly mistyped as a, o, and i; the letter i is most often mistyped as a, u, and e. For the most part, vowel substitutions can be considered to be cognitive errors (except $o \rightarrow i$ may be a cognitive error or typographic error). The effect of keyboard key proximity is also evident; b is often typed as g; d as s; m as n; and so on. Other errors seem to be a result of phonetic similarity; c is misspelled as k and s; q as g and k; and v as w. In general, the edit probabilities roughly match those derived using a corpus of word pairs in (Kernighan, Church, and Gale, 1990).

The insertion probabilities for each letter are shown in Figure 4. Equation 6 is used to convert the edit distances to probabilities. Words in the plural form cause problems for the algorithm, as is illustrated by the high probability of s insertion in Fig-

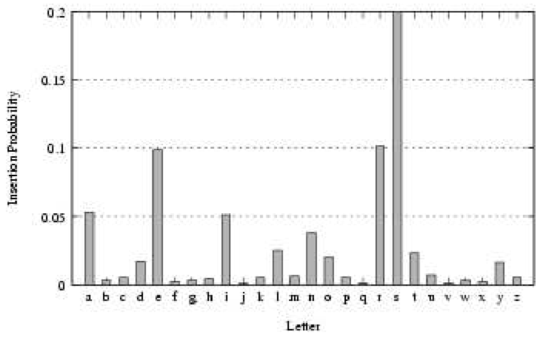


Figure 4: Letter Insertion Probabilities

Spell Checker	ISPELL 3.1.20	ASPELL 0.29	EMBED	Google
Total Tokens	508	508	508	508
Total Found	272 (53.5%)	480 (94.5%)	402 (79.1%)	-
Top 1 (%)	197 (38.8%)	302 (59.5%)	211 (41.5%)	291 (57%)
Top 5 (%)	260 (51.2%)	435 (85.6%)	331 (65.2%)	-
Top 25 (%)	272 (53.5%)	478 (94.1%)	386 (76.0%)	-

Table 4: Spelling Correction Accuracy

ure 4. That is because high frequency query words often appear in both singular and plural form. Every time the singular form is encountered, the plural is considered as a viable candidate correction, and the s insertion probability is increased. Complementarily, every time the plural form is seen, the singular form is considered, increasing the s deletion probability. Indeed, as can be seen in Table 3, deletion is the highest probability operation for the letter s.

5 Testing

To test the accuracy of the error model, the well-known Unix based spelling correction programs *Ispell*[2] and *Aspell*[3] spell checking programs were used for comparison. Ispell generates candidate corrections by considering all words within 1 edit of the misspelled word. *Aspell* uses the *metaphone* algorithm (Philips, 1990), which divides English words into phonetic segments, and generates alternate spellings by substituting similar sounding phones. The test data set[4] consists of 547 misspelled words paired with their best correction as determined by a human expert.Compound words were removed from the test set, leaving 508 misspellings. Several of the misspellings differ from the correction by multiple edit operations. Only the error model learned by the EM algorithm on the search engine queries was used; instead of using the probabilistic language model derived from the query logs and used for training, the word list in */usr/dict/words* was used, with equal probability assigned to each word. Since the test data is composed of single words of varying prevalence, a language model does not significantly aid correction. In practice, the language model would improve performance.

Table 4 compares the performance of the *Aspell* and *Ispell* spell checkers with the Expectation Maximization Based Edit Distance (EMBED) spelling correction system described in this paper. The percentages refer to the percentage of instances in which the correct correction was within the top N suggestions given by the algorithm. If only the top recommended correction is considered, EMBED fares better than *Ispell*, but worse than *Aspell*. For the top 5 and 25 corrections, the rankings of the algorithms are the same.

As Table 4 shows, in several cases the EMBED algorithm did not find the correction within the top 25 suggestions. Typically, the misspellings that could not be found had large edit distances from their corrections. For example, suggestions for the misspelling ”extions” included ”actions” and ”motions” but not the desired correction ”extensions”. In general, by using a phonetic model to compress English words, *Aspell* can find misspellings that have larger edit distances from their correction. However, it relies on a language specific pronunciation model that is manually derived. EM based spelling correction, on the other hand, can be learned from a unlabeled corpus and can be applied to other languages without modification. Although the test data set was comprised of misspelled dictionary words for the purposes of comparison, the spelling correction system described here can handle a continuously evolving vocabulary. Also, the approach described here can be used to train more general error models.

[2]International Ispell Version 3.1.20. http://www.lasr.cs.ucla.edu/geoff/ispell.html

[3]Kevin Atkinson. Aspell Version 0.29. http://aspell.sourceforge.net/

[4]Kevin Atkinson. http://aspell.net/test/

Comparison to online spelling suggestion systems such as provided by the *Google* search engine is difficult since search results are returned for nearly every query on account of the large lexicon. Consequently, many suggestions provided by *Google* are reasonable, but do not correspond to the golden standard in the test data. For example, ”cimplicity” and ”hallo” are not considered misspellings since several online companies and products contain these terms, and ”verison” is corrected to ”verizon” rather than ”version.” While *Google* returns 291 corrections in agreement with the data set (57%), another 44 were judged to be acceptable corrections, giving an accuracy of 66%. In addition, several of the apparently misspelled test strings are new words, proper names, or commonly accepted alternate spellings that are common on the web, so no suggestions were given. Taking these words into account would further improve the accuracy rating.

6 Conclusions and Future Work

The EM algorithm is able to learn an accurate error model without relying on a corpus of paired strings. The edit probabilities determined using the EM algorithm are similar to error models previously generated using other approaches. In addition, the generated error model can be used to find the correct spelling of misspelled words as described in Section 5. However, there are several improvements that can be made to improve spelling error correction. One step is increasing the size of the corpus. While the corpus included nearly 580,000 queries, several thousand of those queries were correctly spelled words without any misspelled versions in the corpus, or misspelled words without the correctly spelled version available. This results in the misidentification of candidate spelling corrections. Another improvement that can improve candidate correction identification is the use of better language models, as discussed in Section 3.2. Since a large proportion of queries contain more than one word, word n-gram statistics can be used to provide context sensitive spelling correction. Finally, a large proportion of typos involve letter transpositions, and other operations that can not be captured by a single-letter substitution model. In (Brill and Moore, 2000), a more general model allowing generic string to string edits is used, allowing many-to-one and one-to-many character substitution edits. Pronunciation modeling in (Toutanova and Moore, 2002) further improves spelling correction performance.

Acknowledgments

Support for this work was provided by the Natural Sciences and Engineering Research Council of Canada.

References

Baeza-Yates, R. 2005. Web Usage Mining in Search Engines. Chapter 14 in *Web Mining: Applications and Techniques*. Ed. Anthony Scime. New York: Idea Group Publishing, 2005. 307-321.

Brill, E. and Cucerzan, S. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. *Proceedings of EMNLP 04*. 293-300.

Brill, E. and Moore, R. 2000. An improved error model for noisy channel spelling correction. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics.* 286 - 293.

Damerau, F. March 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM.* 7(3):171-176.

Kernighan, M., Church, K., and Gale, W. 1990. A spelling correction program based on a noisy channel model. *Proceedings of COLING 1990*. 205-210.

Kucich, K. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*. 24(4):377-439.

Philips, L. 1990. Hanging on the metaphone. *Computer Language Magazine.* 7(12):39.

Ristad, E. and Yianilos, P. 1997. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 20(5):522-532.

Toutanova, K. and Moore, R. 2002. Pronunciation modeling for improved spelling correction. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.* 144-151.

Wagner, R. and Fischer, M. January 1974. The string-to-string correction problem. *Journal of the ACM*. 21(1):168-173.

Wu, S. and Manber, U. 1992. Fast text searching allowing errors. *Communications of the ACM.* 35(10):83-91

Query Expansion with the Minimum User Feedback by Transductive Learning

Masayuki OKABE
Information and Media Center
Toyohashi University of Technology
Aichi, 441-8580, Japan
okabe@imc.tut.ac.jp

Kyoji UMEMURA
Information and Computer Sciences
Toyohashi University of Technology
Aichi, 441-8580, Japan
umemura@tutics.tut.ac.jp

Seiji YAMADA
National Institute of Informatics
Tokyo,101-8430, Japan
seiji@nii.ac.jp

Abstract

Query expansion techniques generally select new query terms from a set of top ranked documents. Although a user's manual judgment of those documents would much help to select good expansion terms, it is difficult to get enough feedback from users in practical situations. In this paper we propose a query expansion technique which performs well even if a user notifies just a relevant document and a non-relevant document. In order to tackle this specific condition, we introduce two refinements to a well-known query expansion technique. One is application of a transductive learning technique in order to increase relevant documents. The other is a modified parameter estimation method which laps the predictions by multiple learning trials and try to differentiate the importance of candidate terms for expansion in relevant documents. Experimental results show that our technique outperforms some traditional query expansion methods in several evaluation measures.

1 Introduction

Query expansion is a simple but very useful technique to improve search performance by adding some terms to an initial query. While many query expansion techniques have been proposed so far, a standard method of performing is to use relevance information from a user (Ruthven, 2003). If we can use more relevant documents in query expansion, the likelihood of selecting query terms achieving high search improvement increases. However it is impractical to expect enough relevance information. Some researchers said that a user usually notifies few relevance feedback or nothing (Dumais and et al., 2003).

In this paper we investigate the potential performance of query expansion under the condition that we can utilize little relevance information, especially we only know a relevant document and a non-relevant document. To overcome the lack of relevance information, we tentatively increase the number of relevant documents by a machine learning technique called *Transductive Learning*. Compared with ordinal inductive learning approach, this learning technique works even if there is few training examples. In our case, we can use many documents in a hit-list, however we know the relevancy of few documents. When applying query expansion, we use those increased documents as if they were true relevant ones. When applying the learning, there occurs some difficult problems of parameter settings. We also try to provide a reasonable resolution for the problems and show the effectiveness of our proposed method in experiments.

The point of our query expansion method is that we focus on the availability of relevance information in practical situations. There are several researches which deal with this problem. Pseudo relevance feedback which assumes top n documents as relevant ones is one example. This method is simple and relatively effective if a search engine returns a hit-list which contains a certain number of relative documents in the upper part. However, unless this assumption holds, it usually gives a worse ranking than the initial search. Thus several researchers propose some specific procedure to make pseudo feedback be effective (Yu and et al, 2003; Lam-Adesina and Jones, 2001). In another way, Onoda (Onoda et al., 2004) tried to apply one-class SVM (Support Vector Machine) to relevance feedback. Their purpose is to improve search performance by using only non-relevant documents. Though their motivation is similar to ours in terms of applying a machine learning method to complement the lack of relevance information, the assumption is somewhat different. Our assumption is to utilizes manual but the minimum relevance judgment.

Transductive leaning has already been applied in the field of image retrieval (He and et al., 2004). In this research, they proposed a transductive method called the manifold-ranking algorithm and showed its effectiveness by comparing with active learning based Support Vector Machine. However, their setting of relevance judgment is not different from many other traditional researches. They fix the total number of images that are marked by a user to 20. As we have already claimed, this setting is not practical because most users feel that 20 is too much for judgment. We think none of research has not yet answered the question. For relevance judgment, most of the researches have adopted either of the following settings. One is the setting of "Enough relevant documents are available", and the other is "No relevant document is available". In contrast to them, we adopt the setting of "Only one relevant document is available". Our aim is to achieve performance improvement with the minimum effort of judging relevancy of documents.

The reminder of this paper is structured as follows. Section 2 describes two fundamental techniques for our query expansion method. Section 3 explains a technique to complement the smallness of manual relevance judgment. Section 4 introduces a whole procedure of our query expansion method step by step. Section 5 shows empirical evidence of the effectiveness of our method compared with two traditional query expansion methods. Section 6 investigates the experimental results more in detail. Finally, Section 7 summarizes our findings.

2 Basic Methods

2.1 Query Expansion

So far, many query expansion techniques have been proposed. While some techniques focus on the domain specific search which prepares expansion terms in advance using some domain specific training documents (Flake and et al, 2002; Oyama and et al, 2001), most of techniques are based on relevance feedback which is given automatically or manually.

In this technique, expansion terms are selected from relevant documents by a scoring function. The Robertson's *wpq* method (Ruthven, 2003) is often used as such a scoring function in many researches (Yu and et al, 2003; Lam-Adesina and Jones, 2001). We also use it as our basic scoring function. It calculates the score of each term by the following formula.

$$wpq_t = \left(\frac{r_t}{R} - \frac{n_t - r_t}{N - R}\right) * \log \frac{r_t/(R - r_t)}{(n_t - r_t)/(N - n_t - R + r_t)} \quad (1)$$

where r_t is the number of seen relevant documents containing term t. n_t is the number of documents containing t. R is the number of seen relevant documents for a query. N is the number of documents in the collection. The second term of this formula is called the Robertson/Spark Jones weight (Robertson, 1990) which is the core of the term weighting function in the Okapi system (Robertson, 1997).

This formula is originated in the following formula.

$$wpq_t = (p_t - q_t) \log \frac{p_t(1 - q_t)}{q_t(1 - p_t)} \quad (2)$$

where p_t is the probability that a term t appears in relevant documents. q_t is the probability that a term t appears in non-relevant documents. We can easily notice that it is very important how the two probability of p_t and q_t should be estimated. The first formula estimates p_t with $\frac{r_t}{R}$ and q_t with $\frac{N_t - R_t}{N - R}$. For the good estimation of p_t and q_t, plenty of relevant document is necessary. Although pseudo feedback which automatically assumes top n documents as relevant is one method and is often used, its performance heavily depends on the quality of an initial search. As we show later, pseudo feedback has limited performance.

We here consider a query expansion technique which uses manual feedback. It is no wonder manual feedback shows excellent and stable performance if enough relevant documents are available, hence the challenge is how it keeps high performance with less amount of manual relevance judgment. In particular, we restrict the manual judgment to the minimum amount, namely only **a relevant document and a non-relevant document**. In this assumption, the problem is how to find more relevant documents based on a relevant document and a non-relevant document. We use transductive learning technique which is suitable for the learning problem where there is small training examples.

2.2 Transductive Learning

Transductive learning is a machine learning technique based on the transduction which directly derives the classification labels of test data without making any approximating function from training data (Vapnik, 1998). Because it does not need to make approximating function, it works well even if the number of training data is small.

The learning task is defined on a data set X of n points. X consists of training data set $L = (\vec{x}_1, \vec{x}_2, ..., \vec{x}_l)$ and test data set $U = (\vec{x}_{l+1}, \vec{x}_{l+2}, ..., \vec{x}_{l+u})$; typically $l \ll u$. The purpose of the learning is to assign a label to each data point in U under the condition that the label of each data point in L are given.

Recently, transductive learning or semi-supervised learning is becoming an attractive subject in the machine learning field. Several algorithms have been proposed so far (Joachims, 1999; Zhu and et al., 2003; Blum and et al., 2004) and they show the advantage of this approach in various learning tasks. In order to apply transductive learning to our query expansion, we select an algorithm called "*Spectral Graph Transducer* (SGT)" (Joachims, 2003), which is one of the state of the art and the best transductive learning algorithms. SGT formalizes the problem of assigning labels to U with an optimization problem of the constrained ratiocut. By solving the relaxed problem, it produces an approximation to the original solution.

When applying SGT to query expansion, X corresponds to a set of top n ranked documents in a hit-list. X does not corresponds to a whole document collection because the number of documents in a collection is too huge[1] for any learning system to process. L corresponds to two documents with manual judgments, a relevant document and a non-relevant document. Furthermore, U corresponds to the documents of $X \cap \overline{L}$ whose relevancy is unknown. SGT is used to produce the relevancy of documents in U. SGT actually assigns values around $\gamma_+ - \theta$ for documents possibly being relevant and $\gamma_- - \theta$ for documents possibly being non-relevant. $\gamma_+ = +\sqrt{\frac{1-f_p}{f_p}}$, $\gamma_- = -\sqrt{\frac{f_p}{1-f_p}}$, $\theta = \frac{1}{2}(\gamma_+ + \gamma_-)$, and f_p is the fraction of relevant documents in X. We cannot know the true value of f_p in advance, thus we have to estimate its approximation value before applying SGT.

According to Joachims, parameter k (the number of k-nearest points of a data $\vec{x}$) and d (the number of eigen values to ...) give large influence to SGT's learning performance. Of course those two parameters should be set carefully. However, besides them, f_p is much more important for our task because it controls the learning performance. Since extremely small L (actually $|L| = 2$ is our setting) give no information to estimate the true value of f_p, we do not strain to estimate its single approximation value but propose a new method to utilize the results of learning with some promising f_p. We describe the method in the next section.

3 Parameter Estimations based on Multiple SGT Predictions

3.1 Sampling for Fraction of Positive Examples

SGT prepares 2 estimation methods to set f_p automatically. One is to estimate from the fraction of positive examples in training examples. This method is not suitable for our task because f_p is always fixed to 0.5 by this method if the number of training examples changes despite the number of relevant documents is small in many practical situations. The other is to estimate with a heuristic that the difference between a setting of f_p and the fraction of positive examples actually assigned by SGT should be as small as possible. The procedure provided by SGT starts from $f_p = 0.5$ and the next f_p is set to the fraction of documents assigned as relevant in the previous SGT trial. It repeats until f_p changes

[1]Normally it is more than ten thousand.

```
Input
  $N_{tr}$   // the number of training examples
Output
  $S$        // a set of sampling points

$piv = \ln(N_{tr})$;  // sampling interval
$nsp = 0$;            // the number of sampling points
for($i = piv$; $i \leq N_{tr} - 1$; $i+ = piv$){
    add $i$ to ;
    nsp++;
    if(nsp == 10){ exit; }
}
```

Figure 1: Pseudo code of sampling procedure for f_p

five times or the difference converges less than 0.01. This method is neither works well because the convergence is not guaranteed at all.

Presetting of f_p is primarily very difficult problem and consequently we take another approach which laps the predictions of multiple SGT trials with some sampled f_p instead of setting a single f_p. This approach leads to represent a relevant document by not a binary value but a real value between 0 and 1. The sampling procedure for f_p is illustrated in Figure 1. In this procedure, sampling interval changes according to the number of training examples. In our preliminary test, the number of sampling points should be around 10. However this number is adhoc one, thus we may need another value for another corpus.

3.2 Modified estimations for p_t and q_t

Once we get a set of sampling points $S = \{f_p^i : i = 1 \sim 10\}$, we run SGT with each f_p^i and laps each resultant of prediction to calculate p_t and q_t as follows.

$$p_t = \frac{\sum_i r_t^i}{\sum_i R_i} \tag{3}$$

$$q_t = \frac{\sum_i n_t - r_t^i}{\sum_i N - R_i} \tag{4}$$

Here, R_i is the number of documents which SGT predicts as relevant with ith value of f_p^i, and r_t^i is the number of documents in R_i where a term t appears. In each trial, SGT predicts the relevancy of documents by binary value of 1 (for relevant) and 0 (for non-relevant), yet by lapping multiple resultant of predictions, the binary prediction value changes to a real value which can represents the relevancy of documents in more detail. The main merit of this approach in comparison with fixing f_p to a single value, it can differentiate a value of p_t if N_{tr} is small.

4 Expansion Procedures

We here explain a whole procedure of our query expansion method step by step.

1. **Initial Search**: A retrieval starts by inputting a query for a topic to an IR system.

2. **Relevance Judgment for Documents in a Hit-List**: The IR system returns a hit-list for the initial query. Then the hit-list is scanned to check whether each document is relevant or non-relevant in descending order of the ranking. In our assumption, this reviewing process terminates when a relevant document and a non-relevant one are found.

3. **Finding more relevant documents by transductive learning**: Because only two judged documents are too few to estimate p_t and q_t correctly, our query expansion tries to increase the number of relevant documents for the *wpq* formula using the SGT transductive learning algorithm. As shown in Figure2, SGT assigns a value of the possibility to be relevant for the topic to each document with no relevance judgment (documents under the dashed line in the Fig) based on two judged documents (documents above the dashed line in the Figure).

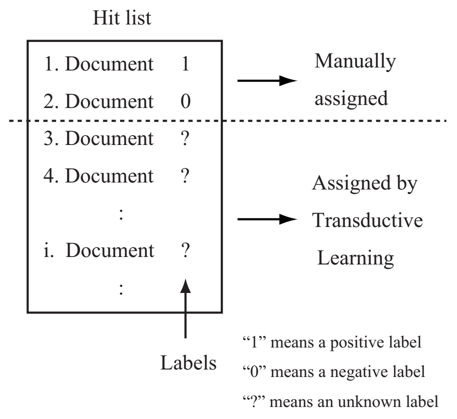

Hit list
1. Document 1
2. Document 0
3. Document ?
4. Document ?
:
i. Document ?
:
Manually assigned
Assigned by Transductive Learning
Labels
"1" means a positive label
"0" means a negative label
"?" means an unknown label

Figure 2: A method to find tentative relevant documents

4. **Selecting terms to expand the initial query**: Our query expansion method calculates the score of each term appearing in relevant documents (including documents judged as relevant by SGT) using *wpq* formula, and then selects a certain number of expansion terms according to the ranking of the score. Selected terms are added to the initial query. Thus an expanded query consists of the initial terms and added terms.

5. **The Next Search with an expanded query**: The expanded query is inputted to the IR system and a new hit-list will be returned. One cycle of query expansion finishes at this step.

   In the above procedures, we naturally introduced transductive learning into query expansion as the effective way in order to automatically find some relevant documents. Thus we do not need to modify a basic query expansion procedure and can fully utilize the potential power of the basic query expansion.

   The computational cost of transductive learning is not so much. Actually transductive learning takes a few seconds to label 100 unlabeled documents and query expansion with all the labeled documents also takes a few seconds. Thus our system can expand queries sufficiently quick in practical applications.

5 Experiments

This section provides empirical evidence on how our query expansion method can improve the performance of information retrieval. We compare our method with other traditional methods.

5.1 Environmental Settings

5.1.1 Data set

We use the TREC-8 data set (Voorhees and Harman, 1999) for our experiment. The document corpus contains about 520,000 news articles. Each document is preprocessed by removing stopwords and stemming. We also use fifty topics (No.401-450) and relevance judgments which are prepared for adhoc task in the TREC-8. Queries for an initial search are nouns extracted from the **title** tag in each topic.

5.1.2 Retrieval Models

We use two representative retrieval models which are bases of the Okapi (Robertson, 1997) and SMART systems. They showed highest performance in the TREC-8 competition.

Okapi : The weight function in Okapi is BM25. It calculates each document's score by the following formula.

$$score(d) = \sum_{T \in Q} w^{(1)} \cdot \frac{(k_1+1)tf(k_3+1)qtf}{(K+tf)(k_3+qtf)} \quad (5)$$

$$w^{(1)} = \log \frac{(r_t+0.5)/(R-r_t+0.5)}{(n_t-r_t+0.5)/(N-n_t-R+r_t+0.5)} \quad (6)$$

$$K = k_1\left((1-b)+b\frac{dl}{avdl}\right) \quad (7)$$

where Q is a query containing terms T, tf is the term's frequency in a document, qtf is the term's frequency in a text from which Q was derived. r_t and n_t are described in section 2. K is calculated by (7), where dl and $avdl$ denote the document length and the average document length. In our experiments, we set $k_1 = 1.2, k_3 = 1000, b = 0.75$, and $avdl = 135.6$. Terms for query expansion are ranked in decreasing order of $r_t \times w^{(1)}$ for the following Okapi's retrieval tests without SGT (**Okapi_manual** and **Okapi_pseudo**) to make conditions the same as of TREC-8.

SMART : The SMART's weighting function is as follows[2].

$$score(d) = \sum_{T \in Q} \{1+\ln(1+\ln(tf))\} * \log(\frac{N+1}{df}) * pivot \quad (8)$$

$$pivot = \frac{1}{0.8+0.2 \times \frac{dl}{avdl}} \quad (9)$$

df is the term's document frequency. tf, dl and $avdl$ are the same as Okapi. When doing relevance feedback, a query vector is modified by the following Rocchio's method (with parameters $\alpha = 3$, $\beta = 2$, $\gamma = 2$).

$$\vec{Q}_{new} = \alpha\vec{Q}_{old} + \frac{\beta}{|D_{rel}|}\sum_{D_{rel}}\vec{d} - \frac{\gamma}{|D_{nrel}|}\sum_{D_{nrel}}\vec{d} \quad (10)$$

[2]In this paper, we use AT&T's method (Singhal et al., 1999) applied in TREC-8

Table 1: Results of Initial Search

	P10	P30	RPREC	MAP	R05P
Okapi_ini	0.466	0.345	0.286	0.239	0.195
SMART_ini	0.460	0.336	0.271	0.229	0.187

D_{rel} and D_{nrel} are sets of seen relevant and non-relevant documents respectively. Terms for query expansion are ranked in decreasing order of the above Rocchio's formula.

Table 1 shows their initial search results of Okapi (**Okapi_ini**) and SMART (**SMART_ini**). We adopt five evaluation measures. Their meanings are as follows (Voorhees and Harman, 1999).

P10 : The precision after the first 10 documents are retrieved.

P30 : The precision after the first 30 documents are retrieved.

R-Prec : The precision after the first R documents are retrieved, where R is the number of relevant documents for the current topic.

MAP : Mean average precision (MAP) is the average precision for a single topic is the mean of the precision obtained after each relevant document is retrieved (using zero as the precision for relevant documents that are not retrieved).

R05P : Recall at the rank where precision first dips below 0.5 (after at least 10 documents have been retrieved).

The performance of query expansion or relevance feedback is usually evaluated on a residual collection where seen documents are removed. However we compare our method with pseudo feedback based ones, thus we do not use residual collection in the following experiments.

5.1.3 Settings of Manual Feedback

For manual feedback, we set an assumption that a user tries to find relevant and non-relevant documents within only top 10 documents in the result of an initial search. If a topic has no relevant document or no non-relevant document in the top 10 documents, we do not apply manual feedback, instead we consider the result of the initial search for

Table 2: Results of **Okapi_sgt** (5 terms expanded)

	P10	P30	RPREC	MAP	R05P
20	0.516	0.381	0.308	0.277	0.233
50	0.494	0.380	0.286	0.265	0.207
100	0.436	0.345	0.283	0.253	0.177

Table 3: Results of **Okapi_sgt** (10 terms expanded)

	P10	P30	RPREC	MAP	R05P
20	0.508	0.383	0.301	0.271	0.216
50	0.520	0.387	0.294	0.273	0.208
100	0.494	0.365	0.283	0.261	0.190

Table 4: Results of **Okapi_sgt** (15 terms expanded)

	P10	P30	RPREC	MAP	R05P
20	0.538	0.381	0.298	0.274	0.223
50	0.528	0.387	0.298	0.283	0.222
100	0.498	0.363	0.280	0.259	0.197

Table 5: Results of **Okapi_sgt** (20 terms expanded)

	P10	P30	RPREC	MAP	R05P
20	0.546	0.387	0.307	0.289	0.235
50	0.520	0.385	0.299	0.282	0.228
100	0.498	0.369	0.272	0.255	0.188

such topics. There are 8 topics [3] which we do not apply manual feedback methods.

5.2 Basic Performance

Firstly, we evaluate the basic performance of our query expansion method by changing the number of training examples. Since our method is based on Okapi model, we represent it as **Okapi_sgt** (with parameters $k = 0.5 * N_{tr}$, $d = 0.8 * N_{tr}$. k is the number of nearest neighbors, d is the number of eigen values to use and N_{tr} is the number of training examples).

Table 2-5 shows five evaluation measures of **Okapi_sgt** when the number of expansion terms changes. We test 20, 50 and 100 as the number of training examples and 5, 10 15 and 20 for the number of expansion terms. As for the number of training examples, performance of 20 and 50 does not differ so much in all the number of expansion terms. However performance of 100 is clearly worse than of 20 and 50. The number of expansion terms does not effect so much in every evaluation measures. In the following experiments, we compare the results of **Okapi_sgt** when the number of training examples is 50 with other query expansion methods.

[3]Topic numbers are 409, 410, 424, 425, 431, 432, 437 and 450

Table 6: Results of Manual Feedback Methods (MAP)

	5	10	15	20
Okapi_sgt	0.265	0.273	0.274	0.282
Okapi_man	0.210	0.189	0.172	0.169
SMART_man	0.209	0.222	0.220	0.219

Table 7: Results of Manual Feedback Methods (10 terms expanded)

	P10	P30	RPREC	MAP	R05P
Okapi_sgt	0.520	0.387	0.294	0.273	0.208
Okapi_man	0.420	0.285	0.212	0.189	0.132
SMART_man	0.434	0.309	0.250	0.222	0.174

Table 8: Results of Pseudo Feedback Methods (MAP)

	5	10	15	20
Okapi_sgt	0.265	0.273	0.274	0.282
Okapi_pse	0.253	0.249	0.247	0.246
SMART_pse	0.236	0.243	0.242	0.242

Table 9: Results of Pseudo Feedback Methods (10 terms expanded)

	P10	P30	RPREC	MAP	R05P
Okapi_sgt	0.520	0.387	0.294	0.273	0.208
Okapi_pse	0.478	0.369	0.279	0.249	0.206
SMART_pse	0.466	0.359	0.272	0.243	0.187

5.3 Comparison with other Manual Feedback Methods

We next compare our query expansion method with the following manual feedback methods.

Okapi_man : This method simply uses only one relevant document judged by hand. This is called *incremental relevance feedback* (Aalbersberg, 1992; Allan, 1996; Iwayama, 2000).

SMART_man : This method is SMART's manual relevance feedback (with parameters $\alpha = 3$, $\beta = 2$, $\gamma = 0$). γ is set to 0 because the performance is terrible if γ is set to 2.

Table 6 shows the mean average precision of three methods when the number of expansion terms changes. Since the number of feedback documents is extremely small, two methods except for **Okapi_sgt** get worse than their initial searches. **Okapi_man** slightly decreases as the number of expansion terms increases. Contrary, **SMART_man** do not change so much as the number of expansion terms increases. Table 7 shows another evaluation measures with 10 terms expanded. It is clear that **Okapi_sgt** outperforms the other two methods.

5.4 Comparison with Pseudo Feedback Methods

We finally compare our query expansion method with the following pseudo feedback methods.

Okapi_pse : This is a pseudo version of Okapi which assumes top 10 documents in the initial search as relevant ones as well as TREC-8 settings.

SMART_pse : This is a pseudo version of SMART. It also assumes top 10 documents as relevant ones. In addition, it assumes top 500-1000 documents as non-relevant ones.

In TREC-8, above two methods uses TREC1-5 disks for query expansion and a phase extraction technique. However we do not adopt these methods in our experiments[4]. Since these methods showed the highest performance in the TREC-8 adhoc task, it is reasonable to compare our method with them as competitors.

Table 8 shows the mean average precision of three methods when the number of expansion terms changes. Performance does not differ so much if the number of expansion terms changes. **Okapi_sgt** outperforms at any number of expansion. Table 9 shows the results in other evaluation measures. **Okapi_sgt** also outperforms except for **R05P**. In particular, performance in **P10** is quite well. It is preferable behavior for the use in practical situations.

6 Discussion

In the experiments, the feedback documents for **Okapi_sgt** is top ranked ones. However some users do not select such documents. They may choose another relevant and non-relevant documents which rank in top 10. Thus we test an another experiment where relevant and non-relevant documents are selected randomly from top 10 rank. Table 10 shows the result. Compared with table 2, the performance seems to become slightly worse. This shows that a

[4]Thus the performance in our experiments is a bit worse than the result of TREC-8

Table 10: Results of **Okapi_sgt** with random feedback (5 terms expanded)

	P10	P30	RPREC	MAP	R05P
20	0.498	0.372	0.288	0.265	0.222
50	0.456	0.359	0.294	0.268	0.200
100	0.452	0.335	0.270	0.246	0.186

user should select higher ranked documents for relevance feedback.

7 Conclusion

In this paper we proposed a novel query expansion method which only use the minimum manual judgment. To complement the lack of relevant documents, this method utilizes the SGT transductive learning algorithm to predict the relevancy of unjudged documents. Since the performance of SGT much depends on an estimation of the fraction of relevant documents, we propose a method to sample some good fraction values. We also propose a method to laps the predictions of multiple SGT trials with above sampled fraction values and try to differentiate the importance of candidate terms for expansion in relevant documents. The experimental results showed our method outperforms other query expansion methods in the evaluations of several criteria.

References

I. J. Aalbersberg. 1992. Incremental relevance feedback. In *Proceedings of SIGIR '92*, pages 11–22.

J. Allan. 1996. Incremental relevance feedback for information filtering. In *Proceedings of SIGIR '96*, pages 270–278.

A. Blum and et al. 2004. Semi-supervised learning using randomized mincuts. In *Proceedings of ICML 2004*.

S. Dumais and et al. 2003. Sigir 2003 workshop report: Implicit measures of user interests and preferences. In *SIGIR Forum*.

G. W. Flake and et al. 2002. Extracting query modification from nonlinear svms. In *Proceedings of WWW 2002*.

J. He and et al. 2004. Manifold-ranking based image retrieval. In *Proceedings of Multimedia 2004*, pages 9–13. ACM.

M. Iwayama. 2000. Relevance feedback with a small number of relevance judgements: Incremental relevance feedback vs. document clustering. In *Proceedings of SIGIR 2000*, pages 10–16.

T. Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML '99*.

T. Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proceedings of ICML 2003*, pages 143–151.

A. M. Lam-Adesina and G. J. F. Jones. 2001. Applying summarization techniques for term selection in relevance feedback. In *Proceedings of SIGIR 2001*, pages 1–9.

T. Onoda, H. Murata, and S. Yamada. 2004. Non-relevance feedback document retrieva. In *Proceedings of CIS 2004*. IEEE.

S. Oyama and et al. 2001. keysword spices: A new method for building domain-specific web search engines. In *Proceedings of IJCAI 2001*.

S. E. Robertson. 1990. On term selection for query expansion. *Journal of Documentation*, 46(4):359–364.

S. E. Robertson. 1997. Overview of the okapi projects. *Journal of the American Society for Information Science*, 53(1):3–7.

I. Ruthven. 2003. Re-examining the potential effectiveness of interactive query expansion. In *Proceedings of SIGIR 2003*, pages 213–220.

A. Singhal, S. Abney, B. Bacchiani, M. Collins, D. Hindle, and F. Pereira. 1999. At&t at trec-8.

V Vapnik. 1998. *Statistical learning theory*. Wiley.

E. Voorhees and D. Harman. 1999. Overview of the eighth text retrieval conference.

S. Yu and et al. 2003. Improving pseud-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of WWW 2003*.

X Zhu and et al. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML 2003*, pages 912–914.

An Orthonormal Basis for Topic Segmentation in Tutorial Dialogue

Andrew Olney
Department of Computer Science
University of Memphis
Memphis, TN 38152
aolney@memphis.edu

Zhiqiang Cai
Institute for Intelligent Systems
University of Memphis
Memphis, TN 38152
zcai@memphis.edu

Abstract

This paper explores the segmentation of tutorial dialogue into cohesive topics. A latent semantic space was created using conversations from human to human tutoring transcripts, allowing cohesion between utterances to be measured using vector similarity. Previous cohesion-based segmentation methods that focus on expository monologue are reapplied to these dialogues to create benchmarks for performance. A novel moving window technique using orthonormal bases of semantic vectors significantly outperforms these benchmarks on this dialogue segmentation task.

1 Introduction

Ever since Morris and Hirst (1991)'s groundbreaking paper, topic segmentation has been a steadily growing research area in computational linguistics, with applications in summarization (Barzilay and Elhadad, 1997), information retrieval (Salton and Allan, 1994), and text understanding (Kozima, 1993). Topic segmentation likewise has multiple educational applications, such as question answering, detecting student initiative, and assessing student answers.

There have been essentially two approaches to topic segmentation in the past. The first of these, lexical cohesion, may be used for either linear segmentation (Morris and Hirst, 1991; Hearst, 1997) or hierarchical segmentation (Yarri, 1997; Choi, 2000). The essential idea behind the lexical cohesion approaches is that different topics will have different vocabularies. Therefore the lexical cohesion within topics will be higher than the lexical cohesion between topics, and gaps in cohesion may mark topic boundaries. The second major approach to topic segmentation looks for distinctive textual or acoustic markers of topic boundaries, e.g. referential noun phrases or pauses (Passonneau and Litman, 1993; Passonneau and Litman, 1997). By using multiple markers and machine learning methods, topic segmentation algorithms may be developed using this second approach that have a higher accuracy than methods using a single marker alone (Passonneau and Litman, 1997).

The primary technique used in previous studies, lexical cohesion, is no stranger to the educational NLP community. Lexical cohesion measured by latent semantic analysis (LSA) (Landauer and Dumais, 1997; Dumais, 1993; Manning and Schütze, 1999) has been used in automated essay grading (Landauer, Foltz, and Laham, 1998) and in understanding student input during tutorial dialogue (Graesser et al., 2001). The present paper investigates an orthonormal basis of LSA vectors, currently used by the AutoTutor ITS to assess student answers (Hu et al., 2003), and how it may be used to segment tutorial dialogue.

The focus on dialogue distinguishes our work from virtually all previous work on topic segmentation: prior studies have focused on monologue rather than dialogue. Without dialogue, previous approaches have only limited relevance to interactive educational applications such as intelligent tutoring systems (ITS). The only existing work on topic segmentation in dialogue, Galley et al. (2003), segments recorded speech between multiple persons using both lexical cohesion and distinctive textual and acoustic markers. The present work differs from Galley et al. (2003) in two respects, viz. we focus solely on textual information and we directly address the problem of tutorial dialogue.

In this study we apply the methods of Foltz et al. (1998), Hearst (1994, 1997), and a new technique utilizing an orthonormal basis to topic segmentation of tutorial dialogue. All three are vector space methods that measure lexical cohesion to determine topic shifts. Our results show that the new using an orthonormal basis significantly outperforms the other methods.

Section 2 reviews previous work, and Section 3 reviews the vector space model. Section 4 introduces an extension of the vector space model which uses an orthonormal basis. Section 5 outlines the task domain of tutorial dialogue, and Section 6 presents the results of previous and the current method on this task domain. A discussion and comparison of these results takes place in Section 7. Section 8 concludes.

2 Previous work

Though the idea of using lexical cohesion to segment text has the advantages of simplicity and intuitive appeal, it lacks a unique implementation. An implementation must define how to represent units of text, compare the cohesion between units, and determine whether the results of comparison indicate a new text segment. Both Hearst (1994, 1997) and Foltz et al. (1998) use vector space methods discussed below to represent and compare units of text. The comparisons can be characterized by a moving window, where successive overlapping comparisons are advanced by one unit of text. However, Hearst (1994, 1997) and Foltz et al. (1998) differ on how text units are defined and on how to interpret the results of a comparison.

The text unit's definition in Hearst (1994, 1997) and Foltz et al. (1998) is generally task dependent, depending on what size gives the best results. For example, when measuring comprehension, Foltz et al. (1998) use the unit of the sentence, as opposed to the more standard unit of the proposition, because LSA is most correlated with comprehension at that level. However, when using LSA to segment text, Foltz et al. (1998) use the paragraph as the unit, to "smooth out" the local changes in cohesion and become more sensitive to more global changes of cohesion. Hearst likewise chooses a large unit, 6 token-sequences of 20 tokens (Hearst, 1994), but varies these parameters dependent on the characteristics of the text to be segmented, e.g. paragraph size.

Under a vector space model, comparisons are performed by calculating the cosine of vectors representing text. As stated previously, these comparisons reflect the cohesion between units of text. In order to use these comparisons to segment text, however, one must have a criterion in place. Foltz et al. (1998), noting mean cosines of .16 for boundaries and .43 for non-boundaries, choose a threshold criterion of .15, which is two standard deviations below the boundary mean of .43. Using LSA and this criterion, Foltz et al. (1998) detected chapter boundaries with an F-measure of .33 (see Manning and Schütze (1999) for a definition of F-measure). Hearst (1994, 1997) in contrast uses a relative comparison of cohesion, by recasting vector comparisons as depth scores. A depth score is computed as the difference between a given vector comparison and its surrounding peaks, i.e. the local maxima of vector comparisons on either side of the given vector comparison. The greater the difference between a given comparison and its surrounding peaks, the higher the depth score. Once all the depth scores are calculated for a text, those that are higher than one standard deviation below the mean are taken as topic boundaries. Using a vector space method without singular value decomposition, Hearst (1997) reports an F-measure of .70 when detecting topic shifts between paragraphs. Thus previous work suggests that the Hearst (1997) method is superior to that of Foltz et al. (1998), having roughly twice the accuracy indicated by F-measure. Although these two results used different data sets and are therefore not directly comparable, one would predict based on this limited evidence that the Hearst algorithm would outperform the Foltz algorithm on other topic segmentation tasks.

3 The vector space model

The vector space model is a statistical technique that represents the similarity between collections of words as a cosine between vectors (Manning and Schütze, 1999). The process begins by collecting text into a corpus. A matrix is created from the corpus, having one row for each unique word in the corpus and one column for each document or paragraph. The cells of the matrix consist of a simple count of the number of times word *i* appeared in document *j*. Since many words do not appear in any given document, the matrix is often sparse. Weightings are applied to the cells that take into account the frequency of word *i* in document *j* and the frequency of word *i* across all documents, such that distinctive words that appear infrequently are given the most weight. Two collections of words of arbitrary size are compared by creating two vectors. Each word is associated with a row vector in the matrix, and the vector of a collection is simply the sum of all the row vectors of words in that collection. Vectors are compared geometrically by the cosine of the angle between them.

LSA (Landauer and Dumais, 1997; Dumais 1993) is an extension of the vector space model that uses singular value decomposition (SVD). SVD is a technique that creates an approximation of the original word by document matrix. After SVD, the original matrix is equal to the product of three matrices, word by singular value, singular value by singular value, and singular value by document. The size of each singular value corresponds to the amount of variance captured by a particular dimension of the matrix. Because the singular values are ordered in decreasing size, it is possible to remove the smaller dimensions and still account for most of the variance. The approximation to the original matrix is optimal, in the least squares sense, for any number of dimensions one would choose. In addition, the removal of smaller dimensions introduces linear dependencies between words that are distinct only in dimensions that account for the least variance. Consequently, two words that were distant in the original space can be near in the compressed space, causing the inductive machine learning and knowledge acquisition effects reported in the literature (Landauer and Dumais, 1997).

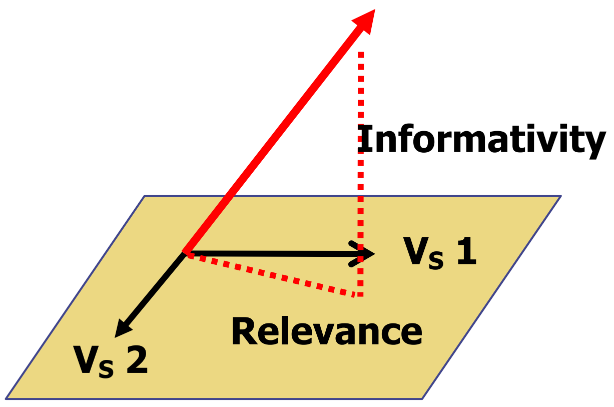


Figure 1. Projecting a new utterance to the basis

4 An orthonormal basis

Cohesion can be measured by comparing the cosines of two successive sentences or paragraphs (Foltz, Kintsch, and Landauer, 1998). However, cohesion is a crude measure: repetitions of a single sentence will be highly cohesive (cosine of 1) even though no new information is introduced. A variation of the LSA algorithm using orthonormalized vectors provides two new measures, "informativity" and "relevance", which can detect how much new information is added and how relevant it is in a context (Hu et al., 2003). The essential idea is to represent context by an orthonormalized basis of vectors, one vector for each utterance. The basis is a subspace of the higher dimensional LSA space, in the same way as a plane or line is a subspace of 3D space. The basis is created by projecting each utterance vector onto the basis of previous utterance vectors using a method known as the Gram-Schmidt process (Anton, 2000). Each projected utterance vector has two components, a component parallel to the basis and a component perpendicular to the basis. These two components represent "informativity" and "relevance", respectively. Let us first consider "relevance". Since each vector in the basis is orthogonal, the basis represents all linear combinations of what has been previously said. Therefore the component of a new utterance vector that is parallel to the basis is already represented by a linear combination of the existing vectors. "Informativity" follows similarly: it is the perpendicular component of a new utterance vector that can not be represented by the existing basis vectors. For example, in Figure 1, a new utterance creates a new vector that can be projected to the basis, forming a triangle. The leg of the triangle that lies along the basis indicates the "relevance" of the recent utterance to the basis; the perpendicular leg indicates new information. Accordingly, a repeated utterance would have complete "relevance" but zero new information.

5 Procedure

The task domain is a subset of conversations from human-human computer mediated tutoring sessions on Newton's Three Laws of Motion, in which tutor and tutee engaged in a chat room-style conversation. The benefits of this task domain are twofold. Firstly, the conversations are already transcribed. Additionally, tutors were instructed to introduce problems using a fixed set of scripted problem statements. Therefore each topic shift corresponds to a distinct problem introduced by the tutor. Clearly this problem would be trivial for a cue phrase based approach, which could learn the finite set of problem introductions. However, the current lexical approach does not have this luxury: words in the problem statements recur throughout the following dialogue.

Human to human computer mediated physics tutoring transcripts first were removed of all markup, translated to lower case, and each utterance was broken into a separate paragraph. An LSA space was made with these paragraphs alone, approximately one megabyte of text. The conversations were then randomly assigned to training (21 conversations) and testing (22 conversations). The average number of utterances per topic, 16 utterances, and the average number of words per utterance, 32 words, were calculated to determine the parameters of the segmentation methods. For example, a moving window size greater than 16 utterances implies that, in the majority of occurrences, the moving window straddles three topics as opposed to the desired two.

To replicate Foltz et al. (1998), software was written in Java that created a moving window of varying sizes on the input text, and the software retrieved the LSA vector and calculated the cosine of each window. Hearst (1994, 1997) was replicated using the JTextTile (Choi, 1999) Java software. A variant of Hearst (1994, 1997) was created by using LSA instead of the standard vector space method. The orthonormal basis method also used a moving window; however, in contrast to the previous methods, the window is not treated just as a large block of text. Instead, the window consists of two orthonormal bases, one on either side of an utterance. That is, a region of utterances above the test utterance is projected, utterance by utterance, into an orthonormal basis, and likewise a region of utterances below the test utterance is projected into another orthonormal basis. Then the test utterance is projected into each orthonormal basis, yielding measures of "relevance" and "informativity" with respect to each. Next the elements that make up each orthonormal basis are aggregated into a block, and a cosine is calculated between the test utterance and the blocks on either side, producing a total of six measures.

Each tutoring session consists of the same 10 problems, discussed between one of a set of 4 tutors and one of 18 subjects. The redundancy provides a variety of speaking and interaction styles on the same topic.

```
Tutor: A clown is riding a
unicycle in a straight line.
She accidentally drops an egg
beside her as she continues
to move with constant veloc-
ity. Where will the egg land
relative to the point where
the unicycle touches the
ground? Explain.
Student: The egg should land
right next to the unicycle.
The egg has a constant hori-
zontal velocity.  The verti-
cal velocity changes and
decreases as gravity pulls
the egg downward at a rate of
9.8m/s^2.  The egg should
therefore land right next to
the unicycle.
Tutor: Good! There is only
one thing I would like to
know. What can you say about
the horizontal velocity of
the egg compared to the hori-
zontal velocity of the clown?
Student: Aren't they the
same?
```

All of the 10 problems are designed to require application of Newton's Laws to be solved, and therefore conversations share many terms such as force, velocity, acceleration, gravity, etc.

6 Results

For each method, the development set was first used to establish the parameters such as text unit size and classification criterion. The methods, tuned to these parameters, were then applied to the testing data.

6.1 Foltz et al. (1998)

In order to replicate Foltz et al.'s results, a text unit size and window size needed to be chosen. The utterance was chosen as the text unit size, which included single word utterances, full sentences, and multi-sentence utterances. To determine the most appropriate window size, results from all sizes between 1 and 16 (the average number of utterances between topic shifts) were gathered. The greatest difference between the means for utterances that introduce a topic shift versus non-shift utterances occurs when the window contains four utterances. The standard deviation is uniformly low for windows containing more than two utterances and therefore can be disregarded in choosing a window size.

The optimal cosine threshold for classification was found using logistic regression (Garson, 2003) which establishes a relationship between the cosine threshold and the log odds of classification. The optimal cutoff was found to be shift odds = .17 with associated F-measure of .49. The logistic equation of best fit is:

$$\ln(\text{shift odds}) = 1.887 + (-13.345 \cdot \text{cosine})$$

F-measure of .49 is 48% higher than the F-measure reported by Foltz et al. (1998) for segmenting monologue. On the testing corpus the F-measure is .52, which demonstrates good generalization for the logistic equation given. Compared the F-measure of .33 reported by Foltz et al. (1998), the current result is 58% higher.

6.2 Hearst (1994, 1997)

The JTextTile software was used to implement Hearst (1994) on dialogue. As with Foltz et al. (1998), a text unit and window size had to be determined for dialogue. Hearst (1994) recommends using the average paragraph size as the window size. Using the development corpus's average topic length of 16 utterances as a reference point, F-measures were calculated for the combinations of window size and text unit size in Table 1.

The optimal combination of parameters (F-measure = .17) is a unit size of 16 words and a window size of 16 units. This combination matches Hearst (1994)'s heuristic of choosing the window size to be the average paragraph length.

		Window size				
		2	**4**	**8**	**16**	**32**
Unit size	**8**	.134	.129	.130	.146	.144
	16	.142	.133	.130	.171	.140
	32	.138	.132	.130	.151	.143

Table 1. Unit vs. window size for Hearst method

On the test set, this combination of parameters yielded an F-measure of .14 as opposed to the F-measure for monologue reported by Hearst (1997), .70. For dialogue, the algorithm is 20% as effective as it is for monologue. It is unclear, however, exactly what part of the algorithm contributes to this poor performance. The two most obvious possibilities are the segmentation criterion, i.e. depth scores, or the standard vector space method.

To further explore these possibilities, the Hearst method was augmented with LSA. Again, the unit size and window size had to be calculated. As with Foltz, the unit size was taken to be the utterance. The window size was determined by computing F-measures on the development corpus for all sizes between 1 and 16. The optimal window size is 9, F-measure = .22. Given the smaller number of test cases, 22, this F-measure of .22 is not significantly different from .17. However, the Foltz method is significantly higher than both of these, $p < .10$.

6.3 Orthonormal basis

The text unit used in the orthonormal basis is the single utterance. The optimal window size, i.e. the orthonormal basis size, was determined by creating a logistic regression to calculate the maximum F-measure for several orthonormal basis sizes. The findings of this procedure are listed in Table 2.

Size	*3*	*4*	*5*	*6*	*8*	*10*	*15*
F	.59	.63	.65	.72	.73	.72	.73

Table 2. F-measure for orthonormal basis sizes

F-measure monotonically increases until the orthonormal basis holds six elements and holds relatively steady for larger orthonormal basis sizes. Since F-measure does not increase much over .72 for greater orthonormal basis sizes, 6 was chosen as the most computationally efficient size for the strength of the effect. The logistic equation of best fit is:

$$\ln(\text{shift odds}) = \begin{aligned} & 20.027 \\ & + (16.703 \cdot \text{cosine}_2) \\ & + (\text{-}30.843 \cdot \text{relevance}_1) \\ & + (\text{-}23.567 \cdot \text{informativity}_1) \\ & + (\text{-}2.698 \cdot \text{relevance}_2) \\ & + (2.771 \cdot \text{informativity}_2) \end{aligned}$$

Where the index of 1 indicates a measure on the window preceding the utterance, and an index of 2 indicates a measure on the window following the utterance. In the regression, the cosine between the utterance and the preceding window was not significant, $p = .86$. This finding reflects the intuition that the cosine to the following window varies according to whether the following window is on a new topic, whereas the cosine to the preceding window is always high. Additionally, measures of “relevance” and “informativity” correspond to vector length; all other measures did not contribute significantly to the model and so were not included.

The sign of the metrics illuminates their role in the model. The negative sign on the coefficients for relevance$_1$, informativity$_1$, and relevance$_2$ indicates that they are inversely correlated with an utterance signaling the start of a new topic. The only surprising feature is that informativity$_1$ is negatively correlated instead of positively correlated: one would expect a topic shift to introduce new information. There is possibly some edge effect here, since the last move of a topic is often a summarizing move that shares many of the physics terms present in the introduction of a new topic. On the other hand, the positive sign on cosine$_2$ and informativity$_2$ indicates that the start of a new topic should have elements in common with the following material and add new information to that material, as an overview would. Beyond the sign, the exponentials of these values indicate how the two basis metrics are weighted. For example, when informativity$_2$ is raised by one unit, a topic shift is 16 times more likely.

On the testing corpus the F-measure of the orthonormal basis method is .67, which is significantly different from the performance of all three methods mentioned above, $p < .05$. Table 3 compares this result with the previous results in the current study for segmenting dialogue.

Method	*Hearst*	*Hearst + LSA*	*Foltz*	*Orth. basis*
F	.14	.22	.52	.67

Table 3. Comparison of dialogue segmentation methods

7 Discussion

The relative ranking of these results is not altogether surprising given the relationships between inferencing and LSA and between inferencing and dialogue. Foltz et al. (1998) found that LSA makes simple bridging inferences in addition to detecting lexical cohesion. These bridging inferences are a kind of collocational cohesion (Halliday and Hassan, 1976) whereby words that cooccur in similar contexts become highly related in the LSA space. Therefore in applications where this kind of inferencing is required, one might expect an LSA based method to excel.

Similarly to van Dijk and Kintsch's model of comprehension (van Dijk and Kintsch, 1983), dialogue can require inferences to maintain coherence. According to Grice's Co-operative Principle, utterances lacking semantic coherence flout the Maxim of Relevance and license an inference (Grice, 1975):

```
S1: Let’s go dancing.
S2: I have an exam tomorrow.
```

The "inference" in the sense of Foltz, Kintsch, and Landauer (1998) would be represented by a high cosine between these utterances, even though they don't share any of the same words. Dialogue generally tends to be less lexically cohesive and require more inferencing than expository monologue, so one might predict that LSA would excel in dialogue applications.

However, LSA has a weakness: the cosine measure between two vectors does not change monotonically as new word vectors are added to either of the two vectors. Accordingly, the addition of a word vector can cause the cosine between two text units to dramatically increase or decrease. Therefore the distinctive properties of individual words can be lost with the addition of more words to a text unit. This problem can be addressed by using an orthonormal basis (Hu et al., 2003). By using a basis, each utterance is kept independent, so "inferencing" can extend over both the entire set of utterances and the linear combination of any of its subsets. Accordingly, when "inferencing" over the entire text unit is required, one would expect a basis method using LSA vectors to outperform a standard LSA method. This expectation has been put to the test recently by Olney & Cai (2005), who find that an orthonormal basis can significantly predict entailment on test data supplied by the PASCAL Textual Entailment Challenge (PASCAL, 2004).

Beyond relative performance rankings, more support for the above reasoning can be found in the difference between Hearst and Hearst + LSA. Recall that in monologue, Hearst (1997) reports a much larger F-measure than Foltz et al. (1998), .70 vs. .33, albeit on different data sets. In the present dialogue corpus, these roles are reversed, .14 vs. .52. Possible reasons for this reversal are the segmentation criterion, the vector space method, or the fact that Foltz has been trained on similar data via regression and Hearst has not. However, comparing the Hearst algorithm with the Hearst + LSA algorithm indicates that a 57% improvement stems from the addition of LSA, keeping all other factors constant. While this result is not statistically significant, the direction of the result supports the use of an "inferencing" vector space method for segmenting dialogue.

Unfortunately, the large difference in F-measure between the Foltz algorithm and the Hearst + LSA algorithm is more difficult to explain. These two methods differ by their segmentation criterion and by their training (Foltz is a regression model and Hearst is not). It may be that Hearst (1994, 1997)'s segmentation criterion, i.e. depth scores, do not translate well to dialogue. Perhaps the assignment of segment boundaries based on the relative difference between a candidate score and its surrounding peaks is highly sensitive to cohesion gaps created by conversational implicatures. On the other hand the differences between these two methods may be entirely attributable to the amount of training they received. One way to separate the contributions of the segmentation criterion and training would be to create a logistic model using the Hearst + LSA method and to compare this to Foltz.

The increased effectiveness of the orthonormal basis method over the Foltz algorithm can also be explained in terms of "inferencing". Since "inferencing" is overwhelmed by lexical cohesion (Foltz et al., 1998), the increase in window size for the Foltz algorithm deteriorates performance for a window size greater than 4. In contrast, the orthonormal basis method becomes most effective as the orthonormal basis size increases past 4. This dichotomy illustrates that the Foltz algorithm is not complementary to an "inferencing" approach in general. Use of an orthonormal basis, on the other hand, increases sensitivity to collocational cohesion without sacrificing lexical cohesion.

8 Conclusion

This study explored the segmentation of tutorial dialogue using techniques that have previously been applied to expository monologue and using a new orthonormal basis technique. The techniques previously applied to monologue reversed their roles of effectiveness when applied to dialogue. This role reversal suggests the predominance of collocational cohesion, requiring "inferencing", present in this tutorial dialogue. The orthonormal basis method, which we suggest has an increased capacity for "inferencing", outperformed both of the techniques previously applied to monologue, and demonstrates that segmentation of these tutorial dialogues most benefits from a method sensitive to lexical and collocational cohesion over large text units.

Acknowledgements

This research was supported by the National Science Foundation (SBR 9720314, REC 0089271, REC 0106965, REC 0126265) and the DoD Multidisciplinary University Research Initiative (MURI) administered by ONR under grant N00014-00-1-0600. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DoD, ONR, or NSF.

References

Anton, H. (2000). Elementary linear algebra. 8th edition. New York: John Wiley.

Barzilay, R. & Elhadad, M. (1997). Using Lexical Chains for Text Summarization. *Proceedings of the Intelligent Scalable Text Summarization Workshop*.

Choi, F. (1999). JTextTile: A free platform independent text segmentation algorithm. http://www.cs.man.ac.uk/~choif

Choi, F. (2000). Advances in domain independent linear text segmentation. In *Proceedings of the NAACL'00*, May.

van Dijk, T. A., & Kintsch, W. (1983). *Strategies of Discourse Comprehension*. New York: Academic Press.

Dumais, S. (1993). LSI meets TREC: a status report. In *Proceedings of the First Text Retrieval Conference* (TREC1), 137-152. NIST Special Publication 500-207.

Foltz, P.W., Kintsch, W. & Landauer, T.K. (1998). The measurement of textual cohesion with latent semantic analysis. *Discourse Processes, 25*, 285-307.

Galley, M., McKeown, K., Fosler-Lussier, E., & Jing, H. (2003). Discourse Segmentation of Multi-Party Conversation. *Proceedings of the ACL.*

Garson, D. Logistic Regression. Accessed on April 18th, 2003. http://www2.chass.ncsu.edu/garson/pa765/logistic

Graesser, A. C., Person, N. K., Harter, D., & the Tutoring Research Group. (2001). Teaching tactics and dialogue in AutoTutor. *International Journal of Artificial Intelligence in Education, 12*, 257-279.

Grice, H.P. (1975). Logic and conversation. In P. Cole & J. Morgan (Eds) *Syntax and Semantics* Vol 3. 41-58. New York: Academic.

Grosz, B.J. & Sidner, C.L. (1986). Attention, Intentions, and the structure of discourse. *Computational Linguistics, 12* (3), 175-204.

Halliday, M. A. & Hassan, R. A. (1976). *Cohesion in English*. London: Longman.

Hearst, M. (1994). Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd meeting of the Association for Computational Linguistics*. 9-16.

Hearst, M. (1997). Text-Tiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics,* 23(1), 33-64.

Hu, X., Cai, Z., Louwerse, M., Olney, A., Penumatsa, P., and Graesser, A. (2003). An improved LSA algorithm to evaluate contributions in student dialogue. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (IJCAI-03), 1489-1491.

Kozima, H. (1993). Text segmentation based on similarity between words. *In Proceedings of ACL '93*, 286-288.

Landauer, T. & Dumais, S. (1997). A solution to Plato 's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review, 104*, 211-240.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. *Discourse Processes, 25*, 259-284.

Manning, C. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press.

Morris, J. & Hirst, G. (1991). Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. *Computational Linguistics, 17*(1), 21-48.

Olney, A., & Cai, Z. (2005). An Orthonormal Basis for Entailment. In Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, 554-559. Menlo Park, Calif.: AAAI Press.

PASCAL. 2004. Recognising Textual Entailment Challenge. Accessed on October 4th, 2004. http://www.pascal-network.org/Challenges/RTE/

Passonneau, R. J. & Litman, D. J. (1993). Intention-based Segmentation: Human Reliability and correlation with linguistic cues. *Proceedings of the ACL*, 148-155.

Passonneau, R. J. & Litman, D. J. (1997). Discourse segmentation by human and automated means. *Computational Linguistics, 23*(1), 103–139.

Salton, G. & Allan, J. (1994). Automatic text decomposition and structuring. *In Proceedings of RIAO*, 6–29, New York, NY.

Yaari, Y. (1997). Segmentation of expository texts by hierarchical agglomerative clustering. *Proceedings of the RANLP'97.*

A Generalized Framework for Revealing Analogous Themes across Related Topics

Zvika Marx
CS and AI Laboratory
MIT
Cambridge, MA 02139, US
zvim@csail.mit.edu

Ido Dagan
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
dagan@cs.biu.ac.il

Eli Shamir
School of Computer Science
The Hebrew University
Jerusalem 91904, Israel
shamir@cs.huji.ac.il

Abstract

This work addresses the task of identifying thematic correspondences across sub-corpora focused on different topics. We introduce an unsupervised algorithmic framework based on distributional data clustering, which generalizes previous initial works on this task. The empirical results reveal interesting commonalities of different religions. We evaluate the results through measuring the overlap of our clusters with clusters compiled manually by experts. The tested variants of our framework are shown to outperform alternative methods applicable to the task.

1 Introduction

The ability to identify analogies and correspondences is one of the fascinating aspects of intelligence. Research in cognitive science has acknowledged the significance of this ability of human thinking, particularly in learning across different situations or domains where the common base to learning is not straightforward. Several previous computational models of analogy making (e.g. Falkenhainer et al., 1989) suggested symbolic computational mechanisms for constructing detailed mappings that connect corresponding ingredients across analogized systems.

This work explores the identification of thematic correspondences in texts through an extension of the well known data clustering problem. Previous works aimed at identifying – through clusters of words – concepts, sub-topics or themes that are prominent within a corpus of texts (e.g., Pereira et al., 1993; Li, 2002; Lin and Pantel, 2002). The current work deals with extending this line of research to identify corresponding themes across a corpus pre-divided to several sub-corpora, which are focused on different, yet related, topics.

This research task has been defined quite recently (Dagan et al., 2002), and has not been explored extensively yet. One could think, however, of many potential applications for drawing correspondences across textual resources: comparison of related firms or products, identifying equivalencies in news published in different countries, and so on. The experimental part of our work deals with revealing correspondences between different religions: Buddhism, Christianity, Hinduism, Islam and Judaism. Given a pre-partition of the corpus to sub-corpora, one for each religion, our method exposes common aspects for all religions, such as *sacred writings*, *festival*s and *suffering*.

The mechanism we employ directs corresponding key terms in the different sub-corpora, such as names of festivals of different religions, to be included in the same cluster. Term clustering methods in general, and in this work in particular, rely on word co-occurrence statistics: terms sharing similar words co-occurrence statistics are clustered together. Different topics, however, are characterized by distinctive terminology and typical word co-locations. Therefore, given a pre-divided corpus, similar co-occurrence patterns would typically be extracted from the same topical sub-corpus. When the terminology and typical phrases employed by each topic differ greatly (even if the topics are essentially related, e.g. different religions), the tendency to form topic-specific clusters intensifies regardless of factors that otherwise could have impact this tendency, such as the co-occurrence window size. Consequently, corresponding key terms of different topics may not be assigned by a standard method to the same cluster, in contrast to our goal. The method described in this paper aims precisely at this problem: it is designed to neutralize salient co-occurrence patterns within each topical sub-corpus and to promote less salient patterns that are shared across the sub-corpora.

In an earlier line of research we have formulated the above problem and addressed it within a probabilistic vector-based setting, presenting two related heuristic algorithms (Dagan et al., 2002; Marx et al., 2004). Here, we devise a general principled distributional clustering paradigm for this problem, termed *cross-partition clustering*, and show that the earlier algorithms are special cases of the new framework.

This paper proceeds as follows: Section 2 describes in more detail the cross-partition clustering problem. Section 3 reviews distributional data clustering methods, which form the basis to our algorithmic framework described in Section 4. Section 5 presents experimental results that reveal interesting themes common to different religions and demonstrates, through an evaluation based on human expert data, that the different variants of our framework outperform alternative methods.

2 The cross-partition clustering problem

The *cross-partition* clustering problem is an extension of the standard (single-set) data clustering problem. In the cross-partition setting, the dataset is pre-partitioned into several distinct *subsets* of elements to be clustered. For example, in our experiments each of these subsets consisted of topical key terms to be clustered. Each such subset was extracted automatically from a sub-corpus corresponding to a different religion (see Section 5).

As in the standard clustering problem, our goal is to cluster the data such that each term cluster would capture a particular theme in the data. However, the generated clusters are expected to identify themes that cut *across* all the given subsets. For example, one cluster consists of names of festivals of different religions, such as *Easter*, *Christmas*, *Sunday* (Christianity) *Ramadan*, *Friday*, *Id-al-fitr* (Islam) and *Sukoth*, *Shavuot*, *Passover* (Judaism; see Figure 4 for more examples).

3 Distributional clustering

Our algorithmic framework elaborates on Pereira et al.'s (1993) *distributional clustering* method. Distributional clustering probabilistically clusters data elements according to the distribution of a given set of features associated with the data. Each data element x is represented as a probability distribution $p(y|x)$ over all features y. In our data $p(y|x)$ is the empirical co-occurrence frequency of a feature word y with a key term x, normalized over all feature word co-occurrences with x.

The distributional clustering algorithmic scheme (Figure 1) is a probabilistic (soft) version of the well-known *K-means* algorithm. It iteratively alternates between:

(1) ***Calculating assignments to clusters***: calculate an assignment probability $p(c|x)$ for each data elements x into each one of the clusters c. This soft assignment is proportional to an information theoretic distance (KL divergence) between the element's $p(y|x)$ representation, and the centroid of c, represented by a distribution $p(y|c)$. The marginal cluster probability $p(c)$ may optionally be set as a prior in this calculation, as in Tishby et al. (1999; in Figure 1 we mark it with dotted underline, to denote it is optional).

Set t = 0, and repeatedly iterate the two update-steps below, till convergence (at time step t = 0, initialize $p_t(c|x)$ randomly or arbitrarily and skip step 1):

(1) $$p_t(c|x) = \underline{p_{t-1}(c)}\frac{e^{-\beta KL[p(y|x)\,||\,p_{t-1}(y|c)]}}{z_t(x,\beta)}$$

where $z_t(x,\beta) = \sum_{c'} \underline{p_{t-1}(c')}e^{-\beta KL[p(y|x)\,||\,p_{t-1}(y|c')]}$

(2) $$p_t(y|c) = \frac{1}{p_t(c)}\sum_x p(x)p_t(c|x)p(y|x)$$

where $p_t(c) = \sum_x p(x)p_t(c|x)$

(3) $$t = t+1$$

Figure 1: A general formulation of the iterative distributional clustering algorithm (with a fixed β value and a fixed number of clusters). The underlined $\underline{p_{t-1}(c)}$ term is optional.

(2) ***Calculating cluster centroids***: calculate a probability distribution $p(y|c)$ over all features y given each cluster c, based on the feature distribution of cluster elements, weighed by the $p(c|x)$ assignment probability calculated in step (1) above. This step imposes the independence of the clusters c of the features y given the data x (similarly to the *naïve Bayes* supervised framework).

Subsequent works (Tishby et al., 1999; Gedeon et al., 2003) have studied and motivated further the earlier distributional clustering method. Particularly, it can be shown that the algorithm of Figure 1 locally minimizes the following *cost function*:

$$F^{dist\text{-}clust} \equiv \underline{H(C)} - H(C|X) + \beta H(Y|C), \qquad (1)$$

where H denotes entropy[1] and X, Y and C are formal variables whose values range over all data elements, features and clusters, respectively.

Tishby et al.'s (1999) *information bottleneck* method (IB) includes the marginal cluster entropy $\underline{H(C)}$ in the cost term[2] (it is marked with dotted underline to denote its inclusion is optional, so that Eq. (1) encapsulates two different cost terms). The addition of $\underline{H(C)}$ corresponds to including the optional prior term $\underline{p_{t-1}(c)}$ in step (1) of the algorithm.

The parameter β that appears in the cost term and in step (1) of the algorithm can have any positive real value. It counterbalances the relative impact of the considerations of maximizing feature information conveyed by the partition to clusters, i.e. minimizing $H(Y|C)$, versus applying the *maximum entropy principle* to the cluster assignment probabilities (see Gedeon et al., 2004), i.e., maximizing $H(C|X)$. The higher β is, the more "determined" the algorithm becomes in assigning each element into the most appropriate cluster. In subsequent runs of the algorithm β can be increased, yielding more separable clusters (clusters with noticeably different centroids) upon convergence. The runs can repeat until, for some β, the desired number of separate clusters is obtained.

4 The cross-partition clustering method

In the cross-partition framework, the pre-partition of the data to subsets is captured through an additional formal variable W, whose values range over the subsets. In our data, each religion corresponds to a different W value, w. Each religion-related key term x is associated with one religion w, with $p(w|x) = 1$ (and $p(w'|x) = 0$ for any $w' \neq w$). Formally, our framework allows probabilistic pre-partition, i.e., $p(w|x)$ values between 0 and 1 but this option was not examined empirically.

The Cross-Partition (CP) clustering method (Figure 2) is an extended version of the probabilistic K-means scheme, introducing additional steps in the iterative loop that incorporate the added pre-partition variable W:

(1) ***Calculating assignments to clusters***, i.e. probabilistic $p(c|x)$ values, is based on current values of cluster *centroids*, as in distributional clustering.

(2) ***Calculating subset-projected cluster centroids***. Given the current element assignments, centroids are computed separately for each combination of

---

Set $t = 0$ and repeatedly iterate the following update steps sequence, till convergence (in the first iteration, when $t = 0$ randomly or arbitrarily initialize $p_t(c|x)$ and skip step CP1):

(1) $p_t(c|x) = \underline{p_{t-1}(c)} \dfrac{e^{-\beta KL[p(y|x) \| p^*_{t-1}(y|c)]}}{z_t(x,\beta)}$

where $z_t(x,\beta) = \sum_{c'} \underline{p_{t-1}(c')} e^{-\beta KL[p(y|x) \| p^*_{t-1}(y|c')]}$

(2) $p_t(y|c,w) = \dfrac{1}{p_t(c,w)} \sum_x p(x) p_t(c|x) p(y|x) p(w|x)$

where $p_t(c,w) = \sum_x p(x) p_t(c|x) p(w|x)$

(3) $p^*_t(c|y) = \underline{p^*_{t-1}(c)} \dfrac{\prod_w p_t(y|c,w)^{\eta p(w)}}{z^*_t(y,\eta)}$

where $z^*_t(y,\eta) = \sum_{c'} \underline{p_{t-1}(c')} \prod_w p_t(y|c',w)^{\eta p(w)}$

(4) $p^*_t(y|c) = \dfrac{1}{p^*_t(c)} \sum_y p(y) p^*_t(c|y)$

where $p^*_t(c) = \sum_y p(y) p^*_t(c|y)$

(5) $t = t + 1$

Figure 2: The cross partition clustering iterative algorithm (with fixed β and η values and a fixed number of clusters). The terms marked by dotted underline, $\underline{p_{t-1}(c)}$ and $\underline{p^*_t(c)}$, are optional.

[1] The entropy of a random variable A is $H(A) = \sum_{a,b} p(a) \log p(a)$, where a ranges over all values of A; the entropy of A conditioned on another variable B is $H(A|B) = \sum_{a,b} p(a,b) \log p(a|b)$, with a and b range over all values of A and B.

[2] The IB cost function was originally formulated as $F^{IB} \equiv I(C;X) - \beta I(C;Y)$. This formulation is equivalent to ours, as $I(C;X) = H(C) - H(C|X)$ and $I(C;Y) = H(Y) - H(Y|C)$, while $H(Y)$ is a constant term depending only on the data.

a cluster c projected on a pre-given subset w. Each such subset-projected centroid is given by a probability distribution $p(y|c,w)$ over the features y, for each c and w separately (instead of $p(y|c)$.

(3) ***Re-evaluating cluster-feature association***. Based on the subset projected centroids, the associations between features and clusters are re-evaluated: features that are commonly prominent across all subsets are promoted relatively to features with varying prominence. A weighted geometric mean scheme achieves this effect: the value of $\prod_w p(y|c,w)^{\eta p(w)}$ is larger as the different $p(y|c,w)$ values are distributed more uniformly over the different w's, for any given c and y. η is a positive valued free parameter, which controls the impact of uniformity versus variability of the averaged values. The re-evaluated associations resulting from this stage are probability distributions over the clusters denoted $p^*(c|y)$. We add an asterisk to distinguish this conditioned probability distribution from other $p(c|y)$ values that can be calculated directly from the output of the previous steps.

(4) ***Calculating cross-partition "global" centroids***: based on the probability distributions $p^*(c|y)$, we calculate a probability distribution $p^*(y|c)$ for every cluster c through a straightforward application of Bayes rule, obtaining the cross partition cluster centroids.

The novelty of the CP algorithm lies in step (3): rather than deriving cluster centroids directly, as in the standard k-means scheme, cluster-feature associations are biased by their prominence *across* the cluster projections over the different subsets. This way, only features that are prominent in the cluster across most subsets end up prominent in the eventual cluster centroid (computed in step 4). By incorporating for every c–y pair a product over all w's, independence of the feature-cluster associations from specific w values is ensured. This conforms to our target of capturing themes that cut across the pre-given partition and are not correlated with specific subsets.

Employing a separate update step in order to accomplish the above direction implies deviation from the familiar cost-based scheme. Indeed, the CP method is not directed by a single cost function that globally quantifies the cross partition clustering task on the whole. Rather, there are four different "local" cost-terms, each articulating a different aspect of the task. As shown in the appendix, each of the update steps (1)–(4) reduces one of these four cost terms, under the assumption that values not modified by that step are held constant. This assumption of course does not hold as values that are not modified by a given step are modified by another. Hence, downward convergence (of any of the cost terms) is not guaranteed.

However, empirical experimentation shows that the dynamics of the CP algorithm tend to stabilize on an equilibrial steady state, where the four different distributions produced by the algorithm balance each other, as illustrated in Figure 3. In fact, convergence occurred in all our text-based experiments (as well as in experiments with synthetic data; Marx et al., 2004).

Manipulating the value of the β parameter works in practice for the CP method as it works for distributional clustering: increasing β along subsequent runs enables the formation of configurations of growing numbers of clusters. The CP framework introduces an additional parameter, η. Intuitively, step (3). As said, the geometric mean scheme promotes those c–y associations for which the $p(y|c,w)$ values are distributed evenly across the w's (for any fixed c and y). A low η would imply a relatively low penalty to those c–y combinations that are not distributed evenly across the w's, but it

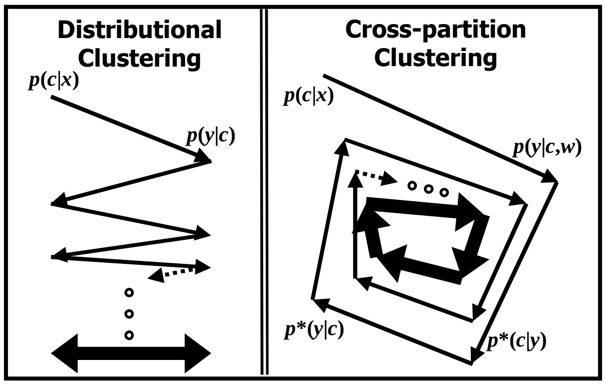


Figure 3: A schematic illustration of the dynamics of the CP framework versus that of distributional clustering. In distributional clustering convergence is onto a configuration where the two systems of distributions complementarily balance one another, bringing a cost term to a locally minimal value. In CP, stable configurations maintain balanced inter-dependencies (*equilibrium*) of four systems of probability distributions.

entails also loss of more information compared to high η. We experimented with η values that are fixed during a whole sequence of runs, while only β is gradually incremented (see Section 5).

Likewise the optional incorporation of priors in the distributional clustering scheme (Figure 1), the CP framework detailed in Figure 2 encapsulates four different algorithmic variants: the prior terms (marked in Figure 2 with dotted underline) can be optionally added in steps (1) and/or (3) of the algorithm. As in the distributional clustering case, the inclusion of these terms corresponds to inclusion of cluster entropy in the corresponding cost terms (see Appendix). It is interesting to note that we introduced previously, on intuitive accounts, some of these variants separately. Here we term the three variations involving priors CP_{I} (prior added in step (1) only, which is the same as the method described in Dagan et al., 2002), CP_{II} (prior added in step (3) only) and CP_{III} (prior added in both steps; as the method in Marx et al., 2004). The version with no priors is denoted CP. Our formulation reveals that these are all special cases of the general CP framework described above.

5 Experimental Results

The data elements that we used for our experiments – religion related key terms – were automatically extracted from a pre-divided corpus addressing five religions: Buddhism, Christianity, Hinduism, Islam and Judaism. The clustered key-term set was pre-partitioned, correspondingly, to five disjoint subsets, one per religion w.[3] In our experimental setting, the key term subsets for the different religions were considered disjoint, i.e., occurrences of the same word in different subsets were considered distinct elements. The set of features y consisted of words that co-occurred with key terms within ±5 word window, truncated by sentence boundaries. About 7000 features, each occurring in all five sub-corpora, were selected.

We survey below some results, which were produced by the plain (unprioired) CP algorithm with $\eta = 0.48$ applied to all five religions together. First, we describe our findings qualitatively and afterwards we provide quantitative evaluation.

[3] We use the dataset of Marx et al. (2004) – five sub-corpora, of roughly one million words each, consisting of introductory web pages, electronic journal papers and encyclopedic entries about the five religions; about 200 key terms were extracted from each sub-corpus to form the clustered subsets.

5.1 Cross-religion Themes

We have found that even the coarsest partition of the data to two clusters was informative and illuminating. It revealed two major aspects that seem to be equally fundamental in the religion domain. We termed them the “spiritual aspect” and “establishment aspect” of Religion. The “spiritual” cluster incorporated terms related with theology, underlying concepts and personal religious experience. Many of the terms assigned to this cluster with highest probability, such as *heaven*, *hell*, *soul*, *god* and *existence*, were in common use of several religions, but it included also religion-specific words such as *atman*, *liberation* and *rebirth* (key concepts of Hinduism). The “establishment” cluster contained names of schools, sects, clergical positions and other terms connected to religious institutions, geo-political entities and so on. Terms assigned to this cluster with high probability were mainly religion specific: *protestant*, *vatican*, *university*, *council* in Christianity; *conservative*, *reconstructionism*, *sephardim*, *ashkenazim* in Judaism and so on (few terms though were common to several religions, for instance *east* and *west*). This two-theme partition was obtained persistently (also when the CP method was applied to pairs of religions rather than to all five). Hence, these aspects appear to be the two universal constituents of religion-related texts in general, to the level the data reflect faithfully this domain.

Clusters of finer granularity still seem to capture fundamental, though more focused, themes. For example, the partition into seven clusters revealed the following topics (our titles): “schools”, “divinity”, “religious experience”, “writings”, “festivals and rite”, “material existence, sin, and suffering” and “family and education”. Figure 4 details the members of highest $p(c|x)$ values within each religion in each of the seven clusters.

The relation between the seven clusters to the coarser two-cluster configuration can be described in soft-hierarchy terms: the “schools” cluster and, to some lesser extent “festivals” and “family”, are related with the “establishment aspect” reflected in the partition to two, while “divinity”, “religious experience” and “suffering” are clearly associated with the “spiritual aspect”. The remaining topic, “writings”, is equally associated with both. The probabilistic framework enabled the CP method to

CCLUSTER 1 “Schools”
Buddhism: america asia japan west east korea india china tibet
Christianity: orthodox protestant catholic west orthodoxy organization rome council america
Hinduism: west christian religious civilization buddhism aryan social founder shaiva
Islam: africa asia west east sunni shiah christian country civilization philosophy
Judaism: reform conservative reconstructionism zionism orthodox america europe sephardim ashkenazim

CLUSTER 2 “Divinity”
Buddhism: god brahma
Christianity: holy-spirit jesus-christ god father savior jesus baptize salvation reign
Hinduism: god brahma
Islam: god allah peace messenger jesus worship believing tawhid command
Judaism: god hashem bless commandment abraham

CLUSTER 3 “Religious Experience”
Buddhism: phenomenon perception consciousness human concentration mindfulness physical liberation
Christianity: moral human humanity spiritual relationship experience expression incarnation divinity
Hinduism: consciousness atman human existence liberation jnana purity sense moksha
Islam: spiritual human physical moral consciousness humanity exist justice life
Judaism: spiritual human existence physical expression humanity experience moral connect

CLUSTER 4 “Writings”
Buddhism: pali-canon sanskrit sutra pitaka english translate chapter abhidhamma book
Christianity: chapter hebrew translate greek new-testament book text old-testament luke
Hinduism: rigveda gita sanskrit upanishad sutra smriti brahma-sutra scripture mahabharata
Islam: chapter surah bible write translate hadith book language scripture
Judaism: tanakh scripture mishnah book oral talmud bible write letter

CLUSTER 5 “Festivals and Rite”
Buddhism: full-moon celebration stupa ceremony sakya abbot ajahn robe retreat
Christianity: easter tabernacle christmas sunday sabbath jerusalem pentecost city season
Hinduism: puja ganesh festival ceremony durga rama pilgrimage rite temple
Islam: kaabah id ramadan friday id-al-fitr haj mecah mosque salah
Judaism: sukoth festival shavuot temple passover jerusalem rosh-hashanah temple-mount rosh-hodesh

CLUSTER 6 “Sin, Suffering, Material Existence”
Buddhism: lamentation water grief kill eat hell animal death heaven
Christianity: fire punishment eat water animal lost hell perish lamb
Hinduism: animal heaven earth death water kill demon birth sun
Islam: water animal hell punishment paradise food pain sin earth
Judaism: animal water eat kosher sin heaven death food forbid

CLUSTER 7 “Family and Education”
Buddhism: child friend son people family question learn hear teacher
Christianity: friend family mother boy question woman problem learn child
Hinduism: child question son mother family learn people teacher teach
Islam: sister husband wife child family marriage mother woman brother
Judaism: child marriage wife mother father women question family people

Figure 4: A sample from a seven-cluster CP configuration of the religion data, including the first members – up to nine – of highest $p(c|x)$ within each religion in each cluster. Cluster titles were assigned by the authors for reference.

cope with these composite relationships between the coarse partition and the finer one.

It is interesting to have a notion of those features y with high $p^*(c|y)$, within each cluster c. We exemplify those typical features, for each one of the seven clusters, through four of the highest $p^*(c|y)$ features (excluding those terms that function as both features and clustered terms):

- “schools” cluster:
  central, *dominant*, *mainstream*, *affiliate*;
- “divinity” cluster:
  omnipotent, *almighty*, *mercy*, *infinite*;
- “religious experience” cluster:
  intrinsic, *mental*, *realm*, *mature*;
- “writings” cluster:
  commentary, *manuscript*, *dictionary*, *grammar*;
- “festivals and rite” cluster:
  annual, *funeral*, *rebuild*, *feast*;
- “material existence, sin, and suffering” cluster:
  vegetable, *insect*, *penalty*, *quench*;
- “community and family” cluster:
  parent, *nursing*, *spouse*, *elderly*.

We demonstratively focus on the two-cluster and seven-cluster, as these numbers are small enough to allow review of all clusters. Configurations of more clusters revealed additional subtopics, such as *education*, *prayer* and so on.

There are some prominent points of correspondence between our findings to Ninian Smart’s comparative religion classics *Dimensions of the Sacred* (1996). For instance, Smart’s *ritual* dimension corresponds to our “festivals and rite” cluster and his *experiential and emotional* dimension corresponds to our “religious experience” cluster.

5.2 Evaluation with Expert Data

We evaluated the performance of our method against cross-religion key term clusters constructed manually by a team of three experts of comparative religion studies. Each manually produced clustering configuration referred to two of the five religions rather than to all five jointly, as in our qualitative review. We examined eight of the ten religion pairs that can be chosen from the total of five. Each religion pair was addressed independently by two different experts using the same set of key terms (so the total number of contributed configurations was 16). Thus, we could also asses the level of agreement between experts.

As an overlap measure we employed the Jaccard coefficient, which is the ratio $n_{11} / (n_{11} + n_{10} + n_{01})$, where:

n_{11} is the number of term pairs assigned to the same cluster by both our method and the expert;

n_{10} is the number of term pairs co-assigned by our method but not by the expert;

n_{01} is the number of term pairs co-assigned by the expert but not by our method.

As the Jaccard score relies on counts of individual term pairs, no assumption with regard to the suitable number of clusters is required. Hence, for each religion pair we produced with our method configurations of two to 16 clusters and calculated for each Jaccard scores based on the overlap with the relevant expert configurations. The scores obtained were averaged over the 15 configurations. The means, over all 16 experimental cases, of those average values are displayed in Table 1.

We tested all four CP method variants, with different fixed values of the η parameter. In addition, we evaluated results obtained by the priored version of distributional clustering (the IB method, Tishby et al., 1999; see Figure 1). Marx et al. (2004) mentioned Information Bottleneck with Side Information (IB-SI, Chechik & Tishby, 2003) as a method capable – unlike standard distributional clustering – of capturing information regarding pre-partition to subsets, which makes this method a seemingly sensible alternative to the CP method. Therefore, we tested the IB-SI method as well, following the adaptation scheme to the CP setting described by Marx et al, with a fixed value of its parameter, $\gamma = 0.07$ (with higher values convergence did not take place in all experiments). As Table 1 shows, the different CP variants performed better than the alternatives. The CP_{III} varinat, with both prior types, was less robust to changes in η value and seemed to be more sensitive to noise.

The experimental part of this work demonstrates that the task of drawing thematic correspondences is challenging. In the particular domain that we have examined the level of agreement between experts seems to make it evident that the task is inherently subjective and just partly consensual. It is remarkable therefore that most variations of our method approximate rather closely the upper bound of the level of agreement between the experts. Further, we have shown the merit of promoting shared cross-subset patterns and neutralizing topic-specific regularities in a newly introduced dedicated computational step. Methods that do not consider this direction (IB) or that incorporate it within a more conventional cost based search (IB-SI) yield notably poorer performance.

Table 1: Mean Jaccard scores for several methods, examined over of the 16 religion-pair evaluation cases (incorporating mean Jaccard scores over 2–16 clustering configurations, see text). The differences between most CP variants and cross-expert agreement are not statistically significant. The differences between IB, IB-SI and CP_{III} with $\eta = 0.83$ and expert agreement are significant (two-tailed t-test, $df = 15$, $p < 0.01$).

	$\eta = 0.48$	$\eta = 0.56$	$\eta = 0.67$	$\eta = 0.83$
CP	0.405	0.383	0.400	0.394
CP_I	0.416	0.400	0.415	0.399
CP_{II}	0. 410	0.387	0.409	0.417
CP_{III}	0. 405	0.420	0.370	0.293
IB: 0.1734	IB-SI ($\gamma = 0.07$): 0.1995			
Agreement between the experts: 0.462				

6 Disscussion

In this paper, we studied and demonstrated the cross partition method, a computational framework that addresses the task of identifying analogies and correspondences in texts. Our approach to this problem bridges between cognitive observations regarding analogy making, which have inspired it, and unsupervised learning techniques.

While previous cognitively-motivated computational frameworks required structured input (e.g. Falkenhainer et al., 1989), the CP method adapts distributional clustering (Pereira et al., 1993), a standard approach applicable to unstructured data. Unlike standard clustering, the CP method considers an additional source of information: pre-partition of the clustered data to several topical subsets (originated in different sub-corpora) between which a correspondence is drawn.

The innovative aspect of the cross-partition method lies in distinguishing feature information that cuts across the given pre-partition to subsets versus subset-specific information. In order to incorporate this aspect within distributional clustering, the CP method interleaves several update steps, each locally optimizing a different cost term.

Our experiments demonstrate that the CP method is capable of revealing interesting and non-trivial corresponding themes in texts. The results obtained with most variants of the CP method, with suitable tuning of the parameters, outperform comparable methods – standard distributional clustering and the IB-SI method – and are rather close to the level of agreement between experts.

The CP method revealed, along various resolution levels, meaningful themes that to our understanding can be considered prominent constituents of Religion. It would be an interesting challenge to apply the CP framework further for other tasks, possibly with more practical flavor, such as comparing and detecting commonalities between commercial products and firms, identifying equivalencies and precedents in legal cases and so on.

References

Gal Chechik and Naftali Tishby. 2003. Extracting relevant structures with side information. In S. Becker, S. Thrun, and K. Obermayer (eds.), *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pp. 857-864.

Ido Dagan, Zvika Marx and Eli Shamir. 2002. Cross-dataset clustering: Revealing corresponding themes across multiple corpora. In D. Roth and A. van den Bosch (eds.), *Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002)*, pp. 15-21.

Brian Falkenhainer, Kenneth D. Forbus and Dedre Gentner. 1989. The structure mapping engine: Algorithm and example. *Artificial Intelligence*, 41(1):1-63.

Tomas Gedeon, Albert E. Parker, and Alexander G. Dimitrov, 2003. Information distortion and neural coding. *Canadian Applied Mathematics Quarterly* 10(1):33-70.

Hang Li. 2002. Word Clustering and Disambiguation based on co-occurrence data, *Natural Language Engineering*, 8(1):25-42.

Zvika Marx, Ido Dagan and Eli Shamir. 2004. Identifying structure across pre-partitioned data. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pp. 489-496.

Dekang Lin and Patrick Pantel. 2002. Concept Discovery from Text. In *Proceedings of Conference on Computational Linguistics (COLING-02)*, pp. 577-583.

Fernando C. Pereira, Nftali Tishby and L. J. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics ACL '93*, pp. 183-190.

Ninian Smart. 1996. *Dimensions of the Sacred: An Anatomy of the World's Beliefs*. University of California Press, Berkeley and Los Angeles, CA.

Naftali Tishby, Fernando C. Pereira and William Bialek. 1999. The information bottleneck method. In *37th Annual Allerton Conference on Communication, Control, and Computing*, pp. 368-379.

Appendix

This appendix specifies the four "local" cost terms mentioned in Section 4 and describes how the CP algorithmic framework (Fig. 2) modifies them.

Step (1) of the CP framework computes $p(c|x)$ values that reduce the value of the following term:

$$F^{CP1} \equiv \underline{H(C)} - H(C|X) + \beta \hat{H}^*(Y|C),$$

where $\hat{H}^*(Y|C) \equiv -\sum_x p(x) \sum_c p(c|x) \sum_y p(y|x) \log p^*(y|c)$. The $p^*(y|c)$ values are considered as if they are constant.

Step (2) computes $p(c|x)$ values reducing the value of

$$F^{CP2} \equiv -\sum_x p(x) \sum_c p(c|x) \sum_y p(y|x) \sum_w p(w|x) \log p(y|c,w),$$

which is equal to $H(Y|C,W)$, subject to an independence assumption extending the assumption explicated in footnote 4, namely for each feature y, cluster c, and pre-given subset w: $p(c,y,w) = \sum_x p(x)\, p(c|x)\, p(y|x)\, p(w|x)$.

Step (3) finds $p^*(c|y)$ values that reduce the value of

$$F^{CP3} \equiv \underline{H^*(C)} - H^*(C|Y) + \eta \hat{H}(Y|C,W),$$

where $H^*(C|Y) = -\sum_y p(y) \sum_c p^*(c|y) \log p^*(c|y)$ and $\hat{H}(Y|C,W) \equiv -\sum_w p(w) \sum_y p(y) \sum_c p^*(c|y) \log p(y|c,w)$, which is equal to the conditioned entropy $H(Y|C,W)$ under an assumption that W is independent of C and Y. The $p(y|c,w)$ values in this term are considered as if they are held constant.

Step (4) finds $p^*(y|c)$ values that reduce the value of

$$F^{CP4} = -\sum_y p(y) \sum_c p^*(c|y) \log p^*(y|c),$$

which can be denoted $H^*(Y|C)$. The $p^*(c|y)$ values are considered as if they are constant.

The underlined $\underline{H(C)}$ and $\underline{H^*(C)}$ terms in F^{CP1} and F^{CP3} are optional; there inclusion implies the inclusion of the prior terms in steps (1) and (3) of the algorithm (see Figure 2).

Flexible Text Segmentation with Structured Multilabel Classification

Ryan McDonald **Koby Crammer** **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{ryantm,crammer,pereira}@cis.upenn.edu

Abstract

Many language processing tasks can be reduced to breaking the text into segments with prescribed properties. Such tasks include sentence splitting, tokenization, named-entity extraction, and chunking. We present a new model of text segmentation based on ideas from multilabel classification. Using this model, we can naturally represent segmentation problems involving overlapping and non-contiguous segments. We evaluate the model on entity extraction and noun-phrase chunking and show that it is more accurate for overlapping and non-contiguous segments, but it still performs well on simpler data sets for which sequential tagging has been the best method.

1 Introduction

Text segmentation is a basic task in language processing, with applications such as tokenization, sentence splitting, named-entity extraction, and chunking. Many parsers, translation systems, and extraction systems rely on such segmentations to accurately process the data. Depending on the application, segments may be tokens, phrases, or sentences. However, in this paper we primarily focus on segmenting sentences into tokens.

The most common approach to text segmentation is to use finite-state sequence tagging models, in which each atomic text element (character or token) is labeled with a tag representing its role in a segmentation. Models of that form include hidden Markov models (Rabiner, 1989; Bikel et al., 1999) as well as discriminative tagging models based on maximum entropy classification (Ratnaparkhi, 1996; McCallum et al., 2000), conditional random fields (Lafferty et al., 2001; Sha and Pereira, 2003), and large-margin techniques (Kudo and Matsumoto, 2001; Taskar et al., 2003). Tagging models are the best previous methods for text segmentation. However, their purely sequential form limits their ability to naturally handle overlapping or non-contiguous segments.

We present here an alternative view of segmentation as *structured multilabel classification*. In this view, a segmentation of a text is a set of *segments*, each of which is defined by the set of text positions that belong to the segment. Thus, a particular segment may not be a set of consecutive positions in the text, and segments may overlap. Given a text $\boldsymbol{x} = x_1 \cdots x_n$, the set of possible segments, which corresponds to the set of possible classification *labels*, is $\text{seg}(\boldsymbol{x}) = \{\texttt{O}, \texttt{I}\}^n$; for $\boldsymbol{y} \in \text{seg}(\boldsymbol{x})$, $y_i = \texttt{I}$ iff x_i belongs to the segment. Then, our segmentation task is to determine which labels are correct segments in a given text. We have thus a structured multilabel classification problem: each instance, a text, may have multiple structured labels, representing each of its segments. These labels are structured in that they do not come from a predefined set, but instead are built from sets of choices associated to the elements of arbitrarily long instances.

More generally, we may be interested in *typed* segments, e.g. segments naming different types of entities. In that case, the set of segment labels is $\mathrm{seg}(\boldsymbol{x}) = T \times \{\mathtt{O}, \mathtt{I}\}^n$, where T is the set of segment types. Since the extension is straightforward, we frame the discussion in terms of untyped segments, and only discuss segment types as needed.

At first sight, it might appear that we have made the segmentation problem intractably harder by turning it into a classification problem with a number of labels exponential on the length of the instance. However, we can bound the number of labels under consideration and take advantage of the structure of labels to find the k most likely labels efficiently. This will allow us to exploit recent advances in online discriminative methods for multilabel classification and ranking (Crammer and Singer, 2002).

Though multilabel classification has been well studied (Schapire and Singer, 1999; Elisseeff and Weston, 2001), as far as we are aware, this is the first study involving structured labels.

2 Segmentation as Tagging

The standard approach to text segmentation is to use tagging techniques with a BIO tag set. Elements in the input text are tagged with one of `B` for the beginning of a contiguous segment, `I` for the inside of a contiguous segment, or `O` for outside a segment. Thus, segments must be contiguous and non-overlapping. For instance, consider the sentence *Estimated volume was a light 2.4 million ounces*. Figure 1a shows how this sentence would be labeled using the BIO tag set for the problem of identifying base NPs in text. Given a particular tagging for a sentence, it is trivial to find all the segments, those whose tag sequences are longest matches for the regular expression $\mathtt{BI}^*$. For typed segments, the BIO tag set is easily augmented to indicate not only segment boundaries, but also the type of each segment. Figure 1b exemplifies the tags for the task of finding people and organizations in text.

Sequential tagging with the BIO tag set has proven quite accurate for shallow parsing and named entity extraction tasks (Kudo and Matsumoto, 2001; Sha and Pereira, 2003; Tjong Kim Sang and De Meulder, 2003). However, this approach can only identify non-overlapping, contiguous segments. This is sufficient for some applications, and in any case, most training data sets are annotated without concern for overlapping or non-contiguous segments. However, there are instances in which sequential labeling techniques using the BIO label set will encounter problems.

Figure 2 shows two simple examples of segmentations involving overlapping, non-contiguous segments. In both cases, it is difficult to see how a sequential tagger could extract the segments correctly. It would be possible to grow the tag set to represent a bounded number of overlapping, non-contiguous segments by representing all possible combinations of segment membership over k overlapping segments, but this would require an arbitrary upper bound on k and would lead to models that generalize poorly and are expensive to train.

Dickinson and Meurers (2005) point out that, as language processing begins to tackle problems in free-word order languages and discourse analysis, annotating and extracting non-contiguous segmentations of text will become increasingly important. Though we focus primarily on entity extraction and NP chunking in this paper, there is no reason why ideas presented here could not be extended to managing other non-contiguous phenomena.

3 Structured Multilabel Classification

As outlined in Section 1, we represent segmentation as multilabel classification, assigning to each text the set of segments it contains. Figure 3 shows the segments for the examples of Figure 2. Each segment is given by a `O`/`I` assignment to its words, indicating which words belong to the segment.

By representing the segmentation problems as multilabel classification, we have fundamentally changed the objective of our learning and inference algorithms. The sequential tagging formulation is aimed to learn and find the best possible tagging of a text. In multilabel classification, we train model parameters so that correct labels — that is, correct segments – receive higher score than all incorrect ones. Likewise, inference becomes the problem of finding the set of correct labels for a text, that is, the set of correct segments.

We now describe the learning problem using the decision-theoretic multilabel classification and ranking framework of Crammer and Singer (2002) and Crammer (2005) as our starting point. In Sec-

```
a. Estimated volume was a light 2.4 million ounces .
      B        I     O B   I    I     I      I    O

b. Bill   Clinton and Microsoft founder Bill   Gates  met today for 20 minutes .
   B-PER   I-PER   O    B-ORG      O    B-PER  I-PER   O    O    O  O     O    O
```

Figure 1: Sequential labeling formulation of text segmentation using the BIO label set. a) NP-chunking tasks. b) Named-entity extraction task.

```
a) Today, Bill and Hilary Clinton traveled to Canada.
   - Person: Bill Clinton
   - Person: Hilary Clinton
b)  ... purified bovine P450 11 beta / 18 / 19 - hydroxylase was ...
   - Enzyme: P450 11 beta-hydroxylase
   - Enzyme: P450 18-hydroxylase
   - Enzyme: P450 19-hydroxilase
```

Figure 2: Examples of overlapping and non-contiguous text segmentations.

tion 3.2, we describe a polynomial-time inference algorithm for finding up to k correct segments.

3.1 Training Multilabel Classifiers

Our model is based on a linear score $s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w})$ for each segment $\boldsymbol{y}$ of text $\boldsymbol{x}$, defined as

$$s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) = \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$$

where $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ is a feature vector representation of the sentence-segment pair, and $\mathbf{w}$ is a vector of feature weights. For a given text $\boldsymbol{x}$, $\text{act}(\boldsymbol{x}) \subseteq \text{seg}(\boldsymbol{x})$ denotes the set of correct segments for $\boldsymbol{x}$, and $\text{best}_k(\boldsymbol{x}; \mathbf{w})$ denotes the set of k segments with highest score relative to the weight vector $\mathbf{w}$. For learning, we use a training set $\mathcal{T} = \{(\boldsymbol{x}_t, \text{act}(\boldsymbol{x}_t))\}_{t=1}^{|\mathcal{T}|}$ of texts labeled with the correct segmentation.

We will discuss later the design of $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ and an efficient algorithm for finding the k highest scoring segments (where k is sufficiently large to include all correct segments). In this section, we present a method for learning a weight vector $\mathbf{w}$ that seeks to score correct segments above all incorrect segments.

Crammer and Singer (2002), extended by Crammer (2005), provide online learning algorithms for multilabel classification and ranking that take one instance at a time, construct a set of scoring constraints for the instance, and adjust the weight vector to satisfy the constraints. The constraints enforce a *margin* between the scores of correct labels and those of incorrect labels. The benefits of large-margin learning are best known from SVMs (Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002), and are analyzed in detail by Crammer (2005) for online multilabel classification.

```
Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \text{act}(\boldsymbol{x}_t))\}_{t=1}^{|\mathcal{T}|}$
1. $\mathbf{w}^{(0)} = 0;\ i = 0$
2. for $n : 1..N$
3.     for $t : 1..|\mathcal{T}|$
4.         $\mathbf{w}^{(i+1)} = \arg\min_{\mathbf{w}} \left\| \mathbf{w} - \mathbf{w}^{(i)} \right\|^2$
           s.t. $s(\boldsymbol{x}_t, \boldsymbol{y}; \mathbf{w}) \geq s(\boldsymbol{x}_t, \boldsymbol{y}'; \mathbf{w}) + 1$
               $\forall \boldsymbol{y} \in \text{act}(\boldsymbol{x}_t), \forall \boldsymbol{y}' \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) - \text{act}(\boldsymbol{x}_t)$
6.         $i = i + 1$
7. $\mathbf{w} = \mathbf{w}^{(N*|\mathcal{T}|)}$
```

Figure 4: A simplified version of the multilabel learning algorithm of Crammer and Singer (2002).

For segmentation, the number of possible labels (segments) is exponential on the length of the text. We make the problem tractable by including only the margin constraints between correct segments and at most k highest scoring incorrect segments. Figure 4 sketches an online learning algorithm for multilabel classification based on the work of Crammer (2005). In the algorithm, $\mathbf{w}^{(i+1)}$ is the projection of $\mathbf{w}^{(i)}$ onto the set of weight vectors such that the scores of correct segments are separated by a margin of at least 1 from the scores of incorrect segments among the k top-scoring segments. This update is *conservative* in that there is no weight change if the constraint set is already satisfied or empty; if some constraints are not satisfied, we make the smallest weight change that satisfies the constraints. Since, the objective is quadratic in $\mathbf{w}$ and the constraints are linear, the optimization problem can be solved by Hildreth's al-

```
a) Today , Bill and Hilary Clinton traveled to Canada .
     O   O   I   O     O       I       O     O     O   O
     O   O   O   O     I       I       O     O     O   O

b)  ... purified bovine P450 11 beta / 18 / 19 - hydroxylase was ...
            O       O     I   I   I  O  O O  O I      I       O
            O       O     I   O   O  O  I O  O I      I       O
            O       O     I   O   O  O  O O  I I      I       O
```

Figure 3: Correct segments for two examples.

gorithm (Censor and Zenios, 1997).

Using standard arguments for linear classifiers (add constant feature, rescale weights) and the fact that all the correct scores in line 4 of Figure 4 are required to be above all the incorrect scores in the top k, that line can be replaced by

$$\begin{array}{l} \mathbf{w}^{(i+1)} = \arg\min_{\mathbf{w}} \left\|\mathbf{w} - \mathbf{w}^{(i)}\right\|^2 \\ \text{s.t. } s(\boldsymbol{x}_t, \boldsymbol{y}; \mathbf{w}) \geq 1 \text{ and } s(\boldsymbol{x}_t, \boldsymbol{y}'; \mathbf{w}) \leq -1 \\ \forall \boldsymbol{y} \in \text{act}(\boldsymbol{x}_t), \forall \boldsymbol{y}' \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) - \text{act}(\boldsymbol{x}_t) \end{array}$$

If v is the number of correct segments for $\boldsymbol{x}$, this transformation replaces $O(kv)$ constraints with $O(k+v)$ constraints: segment scores are compared to a single positive or negative threshold rather then to each other. At test time, we find the segments with positive score by finding the k highest scoring segments and discarding those with a negative score.

3.2 Inference

During learning and at test time we require a method for finding the k highest scoring segments. At test time, we predict as correct all the segments with positive score in the top k. In this section we give an algorithm that calculates this precisely.

For inference, tagging models typically use the Viterbi algorithm (Rabiner, 1989). The algorithm is given by the following standard recurrences:

$$\begin{array}{rcl} S[i,t] & = & \max_{t'} s(t',t,i) + S[i-1,t'] \\ B[i,t] & = & \arg\max_{t'} s(t',t,i) + S[i-1,t'] \end{array}$$

with appropriate initial conditions, where $s(t',t,i)$ is the score for going from tag t' at $i-1$ to tag t at i. The dynamic programming table $S[i,t]$ stores the score of the best tag sequence ending at position i with tag t, and $B[i,t]$ is a back-pointer to the previous tag in the best sequence ending at i with t, which allows us to reconstruct the best sequence. The Viterbi algorithm has easy k-best extensions.

We could find the k highest scoring segments using Viterbi. However, for the case of non-contiguous segments, we would like to represent higher-order dependencies that are difficult to model in Viterbi. In particular, in Figure 3b we definitely want a feature bridging the gap between *Bill* and *Clinton*, which could not be captured with a standard first-order model. But moving to higher-order models would require adding dimensions to the dynamic programming tables S and B, with corresponding multipliers to the complexity of inference.

To represent dependencies between non-contiguous text positions, for any given segment $\boldsymbol{y} = y_1 \cdots y_n$, let $i(\boldsymbol{y}) = 0 i_1 \cdots i_m (n+1)$ be the increasing sequence of indices i_j such that $y_{i_j} = \mathtt{I}$, padded for convenience with the dummy first index 0 and last index $n+1$. Also for convenience, set $x_0 = $ `-s-` and $x_{n+1} = $ `-e-` for fixed start and end markers. Then, we restrict ourselves to feature functions $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ that factor relative to the input as

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{|i(\boldsymbol{y})|} \mathbf{g}(i(\boldsymbol{y})_{j-1}, i(\boldsymbol{y})_j) \qquad (1)$$

where $i(\boldsymbol{y})_j$ is the j^{th} integer in $i(\boldsymbol{y})$ and $\mathbf{g}$ is a feature function depending on arbitrary properties of the input relative to the indices $i(\boldsymbol{y})_{j-1}$ and $i(\boldsymbol{y})_j$.

Applying (1) to the segment *Bill Clinton* in Figure 3, its score would be

$$\mathbf{w} \cdot [\mathbf{g}(0,3) + \mathbf{g}(3,6) + \mathbf{g}(6,11)]$$

This feature representation allows us to include dependencies between non-contiguous segment positions, as well as dependencies on any properties of the input, including properties of skipped positions.

We now define the following dynamic program

$$\begin{array}{rcl} S[i] & = & \max_{j<i} S[j] + \mathbf{w} \cdot \mathbf{g}(j,i) \\ B[i] & = & \arg\max_{j<i} S[j] + \mathbf{w} \cdot \mathbf{g}(j,i) \end{array}$$

These recurrences compute the score $S[i]$ of the best partial segment ending at i as the sum of the maximum score of a partial segment ending at position $j < i$, and the score of skipping from j to i. The back-pointer table B allows us to reconstruct the sequence of positions included in the segment.

Clearly, this program requires $O(n^2)$ time for a text of length n. Furthermore we can easily augment this algorithm in the standard fashion to find the k best segments, and multiple segment types, resulting in a runtime of $O(n^2kT)$, where T is the number of types. $O(n^2kT)$ is not ideal, but is still practical since in this work we are segmenting sentences. If we can bound the largest gap in any non-contiguous segment by a constant $g \ll n$, then the runtime can be improved to $O(ngkT)$. This runtime does not compare favorably to the standard Viterbi algorithm that runs in $O(nT^2)$, especially for large k. However, we found that for even large k we could still train large models in a matter of hours and test on unseen data in a few minutes.

3.2.1 Restrictions

Often a segmentation task or data set will restrict particular kinds of segments. For instance, it may be the case that a data set does not have any overlapping or non-contiguous segments. Embedded segmentations – those in which one segment's tokens are a subset of another's – is also a phenomenon that sometimes does not occur.

It is easy to restrict the inference algorithm to disallow such segments if they are unnecessary. For example, if two segments overlap or are embedded, the inference algorithm can just return the highest scoring one. Or it can simply ignore all non-contiguous segments if it is known that they do not occur in the data. In Section 4 we will augment the inference algorithm accordingly for each data set.

3.3 Feature Representation

We now discuss the design of the feature function for two consecutive segment positions $\mathbf{g}(j, i)$, where $j < i$. We build individual binary-valued features from predicates over the input, for instance, the identities of words in the sentence at particular positions relative to i and j. The selection of predicates varies by task, and we provide specific predicate sets in Section 4 for various data sets. In this section, we use for illustration word-pair identity predicates such as $x_j = \texttt{Bill}\ \&\ x_i = \texttt{Clinton}$.

For sequential tagging models, predicates are combined with the set of states (or tags) to create a feature representation. For our model, we define the following possible states:

$$\begin{array}{rcl} \text{start} & \equiv & j = 0 \\ \text{end} & \equiv & i = n + 1 \\ \text{next} & \equiv & j = i - 1 \\ \text{skip} & \equiv & j < i - 1 \end{array}$$

For example, the following features would be on for $\mathbf{g}(0, 3)$[1] and $\mathbf{g}(3, 6)$, respectively, in Figure 3a:

$$x_j = \texttt{-s-}\ \&\ x_i = \texttt{Bill}\ \&\ \text{start}$$
$$x_j = \texttt{Bill}\ \&\ x_i = \texttt{Clinton}\ \&\ \text{skip}$$

These features indicate a predicate's role in the segment: at the beginning, at the end, over contiguous segment words or skipping over some words. All features can be augmented to indicate specific segment types for multi-type segmentation tasks. No matter what the task, we always add predicates that represent ranges of the distance $i - j$, as well as what words or part-of-speech tags occur between the two words. For instance, $\mathbf{g}(3, 6)$ might contain

$$\texttt{word-in-between} = \texttt{and}\ \&\ \text{skip}$$

These features are designed to identify common characteristics of non-contiguous segments such as the presence of conjunctions or punctuation in skipped portions. Although we have considered only binary features here, the model in principle allows arbitrary real-valued feature.

3.4 Summary

We presented a method for text segmentation that equates the problem to structured multilabel classification where each label corresponds to a segment. We showed that learning and inference can be managed tractably in the formulation by efficiently finding the k highest scoring segments through a dynamic programming algorithm that factors the structure of each segment. The only concern is that k must be large enough to include all correct segments,

[1]Note that "skip" is not on for $\mathbf{g}(0, 3)$ even though $j < i-1$. Start and end states override other states.

which we will discuss further in Section 4. This method naturally models all possible segmentations including those with overlapping or non-contiguous segments. Out approach can be seen as multilabel variant of the work of McDonald et al. (2004), which creates a set of constraints to separate the score of the single correct output from the k highest scoring outputs with an appropriate large margin.

4 Experiments

We now describe a set of experiments on named entity and base NP segmentation. For these experiments, we set $k = n$, where n is the length of the sentence. This represents a reasonable upper bound on the number of entities or chunks in a sentence and results in a time complexity of $O(n^3T)$.

We compare our methods with both the averaged perceptron (Collins, 2002) and conditional random fields (Lafferty et al., 2001) using identical predicate sets. Though all systems use identical predicates, the actual features of the systems are different due to the fundamental differences between the multilabel classification and sequential tagging models.

4.1 Standard data sets

Our first experiments are standard named entity and base NP data sets with no overlapping, embedded or non-contiguous segments. These experiments will show that, for simple segmentations, our model is competitive with sequential tagging models.

For the named entity experiments we used the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) data with people, organizations, locations and miscellaneous entities. We used standard predicates based on word, POS and orthographic information over a previous to next word window. For the NP chunking experiments we used the standard CoNLL 2000 data set (Kudo and Matsumoto, 2001; Sha and Pereira, 2003) using the predicate set defined by Sha and Pereira (2003).

The first three rows of Table 1 compare the multilabel classification approach to standard sequential classifiers. As one might expect, the performance of the multilabel classification method is below that of the sequential tagging methods. This is because those methods model contiguous segments well without the need for thresholds or k-best inference. In addition, the multilabel method shows significantly higher precision then recall. One possible reason for this is that during the course of learning, the model will see many segments that are *nearly* correct, e.g., segments that overlap correct segments and differ by a single token. As a result, the model learns to score all segments containing even a small amount of negative evidence as invalid in order to ensure that these nearly correct segments have a sufficiently low score.

One way to alleviate this problem is to restrict the inference algorithm to not return any overlapping, non-contiguous or embedded segmentations as discussed in Section 3.2.1, since this data set does not contain segments of this kind. This way, the learning stage only updates the parameters when a *nearly* correct segment actually out scores the correct one. The results of this system are shown in row 4 of Table 1. We can see that this change did lead to a more balanced precision/recall, however it is clear that more investigation is required.

4.2 Chemical substance extraction

The second set of experiments involves extracting chemical substance names from MEDLINE abstracts that relevant to the inhibition of the enzyme CYP450 (PennBioIE, 2005). We focus on abstracts that have at least one overlapping or non-contiguous annotation. This data set contains 6164 annotated chemical substances, including 6% that are both overlapping and non-contiguous. Figure 3b is an example from the corpus. We use identical predicates to the named entity experiments in Section 4.1. Though the data does contain overlapping and non-contiguous segments, it does not contain embedded segments. Results are shown in Table 2 using 10-fold cross validation. The sequential tagging models were trained using only sentences with no overlapping or non-contiguous entities. We found this provided the best performance. Row 4 of Table 2 shows the multilabel approach with the inference algorithm restricted to not allow embedded segments.

We can see that our method does significantly better on this data set (up to a 26% reduction in error). It is also apparent that the model is picking up some overlapping and non-contiguous entities (see Table 2). However, the models performance on these kinds of entities is lower than overall performance.

	a. **Named-Entity Extraction**			b. **NP-chunking**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
Avg. Perceptron	82.46	83.14	82.80	94.22	93.88	94.05
CRFs	83.36	**83.57**	**83.47**	94.57	**94.00**	**94.29**
Multilabel	**92.47**	74.19	82.33	**94.65**	92.28	93.45
Multilabel with Restrictions	91.08	76.68	83.26	94.10	93.70	93.90

Table 1: Results for named-entity extraction and NP-chunking on data sets with only non-overlapping and contiguous segments annotated.

	Chem Substance Extraction - A			**Chem Substance Extraction - B**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
Avg. Perceptron	82.98	79.40	81.15	**1.0**	0.0	0.0
CRFs	85.85	79.06	82.31	**1.0**	0.0	0.0
Multilabel	88.24	80.84	84.38	62.56	33.67	43.78
Multilabel with Restrictions	**88.55**	**84.59**	**86.53**	72.58	**45.92**	**56.25**

Table 2: Results for chemical substance extraction. Table A is for all entities in the data set and Table B is only for those entities that are overlapping and non-contiguous.

4.3 Tuning Precision and Recall

The learning algorithm in Section 3.1 seeks a separator through the origin, though, our experimental results suggest that this tends to favor precision at the expense of recall. However, at test time we can use a separation threshold different from zero. This parameter allows us to trade off precision against recall, and could be tuned on held-out data.

Figure 5 plots precision, recall and f-measure against the threshold for the basic multilabel model on the chemical substance, NP chunking and person entity extraction data sets. These plots clearly show what is expected: higher thresholds give higher precision, and lower thresholds give higher recall. In these data sets at least, a zero threshold is almost always near optimal, though sometimes we would benefit from a slightly lower threshold.

5 Discussion

We have presented a method for text segmentation that is base on discriminatively learning structured multilabel classifications. The benefits include

- Competitive performance with sequential tagging models.
- Flexible modeling of complex segmentations, including overlapping, embedded and non-contiguous segments.
- Adjustable precision-recall trade off.

However, there is a computation cost for our models. For a text of length n, training and testing require $O(n^3 T)$ time, where T is the number of segment types. Fortunately, this still results in training times on the order of hours.

Our approach is related to the work of Bockhorst and Craven (2004). In this work, a conditional random field model is trained to allow for overlapping segments with an $O(n^2)$ inference algorithm. The model is applied to biological sequence modeling with promising results. However, our approaches differ in two major respects. First, their model is probabilistic, and trained to maximize segmentation likelihood, while our model is trained to maximize margin. Second, our method allows for non-contiguous segments, at the cost of a slower $O(n^3)$ inference algorithm.

In further work, the classification threshold should also be learned to achieve the desired balance between precision and recall. It would also be useful to investigate methods for combining these models with standard sequential tagging models to get top performance on simple segmentations as well as on overlapping or non-contiguous ones.

A broader area of investigation are other problems in language processing that can benefit from structured multilabel classification, e.g., ambiguities in language often result in multiple acceptable parses for sentences. It may be possible to extend the algorithms presented here to learn to distinguish all acceptable parses from unacceptable ones instead of just finding a single parse when many are valid.

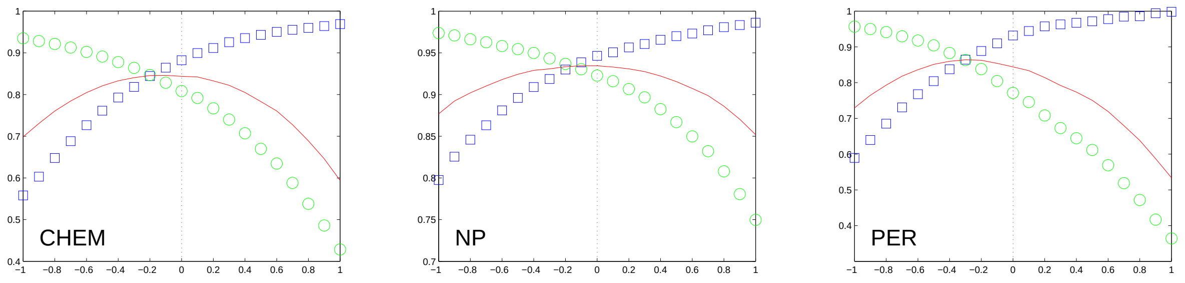


Figure 5: Precision (squares), Recall (circles) and F-measure (line) plotted against threshold values. CHEM: chemical substance extraction, NP: noun-phrase chunking, and PER: person name extraction.

Acknowledgments

We thank the members of the Penn BioIE project for the development of the CYP450 corpus that we used for our experiments. In particular, Seth Kulick answered many questions about the data. This work has been supported by the NSF ITR grant 0205448.

References

D.M. Bikel, R. Schwartz, and R.M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34(1/3):221–231.

J. Bockhorst and M. Craven. 2004. Markov networks for detecting overlapping elements in sequence data. In *Proc. NIPS*.

Y. Censor and S.A. Zenios. 1997. *Parallel optimization : theory, algorithms, and applications*. Oxford University Press.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

K. Crammer and Y. Singer. 2002. A new family of online algorithms for category ranking. In *Proc SIGIR*.

K. Crammer. 2005. *Online Learning for Complex Categorial Problems*. Ph.D. thesis, Hebrew University of Jerusalem. to appear.

N. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

M. Dickinson and W.D. Meurers. 2005. Detecting errors in discontinuous structural annotation. In *Proc. ACL*.

A. Elisseeff and J. Weston. 2001. A kernel method for multi-labeled classifi cation. In *Proc. NIPS*.

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proc. NAACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fi elds: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML*.

R. McDonald, K. Crammer, and F. Pereira. 2004. Large margin online learning algorithms for scalable structured classication. In *NIPS Workshop on Structured Outputs*.

PennBioIE. 2005. Mining The Bibliome Project. http://bioie.ldc.upenn.edu/.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. EMNLP*.

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confi dence-rated predictions. *Machine Learning*, 37(3):1–40.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fi elds. In *Proc. HLT-NAACL*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proc. NIPS*.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*.
http://www.cnts.ua.ac.be/conll2003/ner.

The Vocal Joystick: A Voice-Based Human-Computer Interface for Individuals with Motor Impairments[*]

Jeff A. Bilmes[†], _Xiao Li_[†], _Jonathan Malkin_[†], _Kelley Kilanski_[‡], _Richard Wright_[‡], _Katrin Kirchhoff_[†], _Amarnag Subramanya_[†], _Susumu Harada_[§], _James A. Landay_[§], _Patricia Dowden_[¶], _Howard Chizeck_[†]

[†]Dept. of Electrical Engineering **[‡]Dept. of Linguistics**
[§]Dept. of Computer Science & Eng. **[¶]Dept. of Speech & Hearing Science**
University of Washington
Seattle, WA

Abstract

We present a novel voice-based human-computer interface designed to enable individuals with motor impairments to use vocal parameters for continuous control tasks. Since discrete spoken commands are ill-suited to such tasks, our interface exploits a large set of continuous acoustic-phonetic parameters like pitch, loudness, vowel quality, etc. Their selection is optimized with respect to automatic recognizability, communication bandwidth, learnability, suitability, and ease of use. Parameters are extracted in real time, transformed via adaptation and acceleration, and converted into continuous control signals. This paper describes the basic engine, prototype applications (in particular, voice-based web browsing and a controlled trajectory-following task), and initial user studies confirming the feasibility of this technology.

1 Introduction

Many existing human-computer interfaces (e.g., mouse and keyboard, touch screens, pen tablets, etc.) are ill-suited to individuals with motor impairments. Specialized (and often expensive) human-computer interfaces that have been developed specifically for this target group include sip and puff switches, head mice, eye-gaze devices, chin joysticks and tongue switches. While many individuals with motor impairments have complete use of their vocal system, these devices make little use of it. Sip and puff switches, for example, have low communication bandwidth, making it impossible to achieve more complex control tasks.

Natural spoken language is often regarded as the obvious choice for a human-computer interface. However, despite significant research efforts in automatic speech recognition (ASR) (Huang et al., 2001), existing ASR systems are still not sufficiently robust to a wide variety of speaking conditions, noise, accented speakers, etc. ASR-based interfaces are therefore often abandoned by users after a short initial trial period. In addition, natural speech is optimal for communication between humans but sub-optimal for manipulating computers, windows-icons-mouse-pointer (WIMP) interfaces, or other electro-mechanical devices (such as a prosthetic robotic arm). Standard spoken language commands are useful for discrete but not for continuous operations. For example, in order to move a cursor from the bottom-left to the upper-right of a screen, the user might have to repeatedly utter “up” and “right” or “stop” and “go” after setting an initial trajectory and rate, which is quite inefficient. For these reasons, we are developing alternative and reusable voice-based assistive technology termed the “Vocal Joystick” (VJ).

2 The Vocal Joystick

The VJ approach has three main characteristics:
1) Continuous control parameters: Unlike standard speech recognition, the VJ engine exploits continuous vocal characteristics that go beyond simple sequences of discrete speech sounds (such as syllables or words) and include e.g., pitch, vowel quality, and loudness, which are then mapped to continuous con-

[*]This material is based on work supported by the National Science Foundation under grant IIS-0326382.

trol parameters.

2) Discrete vocal commands: Unlike natural speech, the VJ discrete input language is based on a pre-designed set of sounds. These sounds are selected with respect to acoustic discriminability (maximizing recognizer accuracy), pronounceability (reducing potential vocal strain), mnemonic characteristics (reducing cognitive load), robustness to environmental noise, and application appropriateness.

3) Reusable infrastructure: Our goal is not to create a single application but to provide a modular library that can be incorporated by developers into a variety of applications that can be controlled by voice. The VJ technology is not meant to replace standard ASR but to enhance and be compatible with it.

2.1 Vocal Characteristics

Three continuous vocal characteristics are extracted by the VJ engine: *energy*, *pitch*, and *vowel quality*, yielding four specifiable continuous degrees of freedom. The first of these, localized acoustic *energy*, is used for voice activity detection. In addition, it is normalized relative to the current vowel detected (see Section 3.3), and is used by our current VJ-WIMP application (Section 4) to control the velocity of cursor movements. For example, a loud voice causes a faster movement than does a quiet voice. The second parameter, *pitch*, is also extracted but is currently not mapped to any control dimension in the VJ-WIMP application but will be in the future. The third parameter is *vowel quality*. Unlike consonants, which are characterized by a greater degree of constriction in the vocal tract, vowels have much inherent signal energy and are therefore well-suited to environments where both high accuracy and noise-robustness are crucial. Vowels can be characterized using a 2-D space parameterized by F1 and F2, the first and second vocal-tract formants (resonant frequencies). We initially experimented with directly extracting F1/F2 and using them for directly specifying 2-D continuous control. While we have not ruled out the use of F1/F2 in the future, we have so far found that even the best F1/F2 detection algorithms available are not yet accurate enough for precise real-time specification of movement. Therefore, we classify vowels directly and map them onto the 2-D vowel space characterized by degree of constriction (i.e., tongue height) and tongue body position (Figure 1). In our VJ-WIMP application, we use

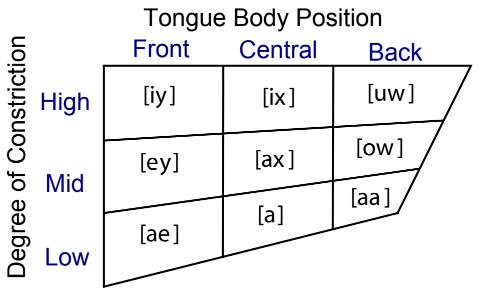


Figure 1: Vowel configurations as a function of their dominant articulatory configurations.

the four corners of this chart to map to the 4 principle directions of up, down, left, and right as shown in Figure 2 (note that the two figures are flipped and rotated with respect to each other). We have four different VJ systems running: A) a *4-class system* allowing only the specification of the 4 principle directions; B) a *5-class system* that also includes the phone [ax] to act as a carrier when wishing to vary only pitch and loudness; C) a *8-class* system that includes the four diagonal directions; and D) a *9-class* system that includes all phones and directions. Most of the discussion in this paper refers to the 4-class system.

A fourth vocal characteristic is also extracted by the VJ engine, namely *discrete sounds*. These sounds may correspond to button presses as on a mouse or joystick. The choice of sounds depends on the application and are chosen according to characteristic 2 above.

3 The VJ Engine

Our system-level design goals are modularity, low latency, and maximal computational efficiency. For this reason, we share common signal processing operations in multiple signal extraction modules, which yields real-time performance but leaves considerable computational headroom for the back-end applications being driven by the VJ engine.

Figure 3 shows the VJ engine architecture having three modules: signal processing, pattern recognition, and motion control.

3.1 Signal Processing

The goal of the signal processing module is to extract low-level acoustic features that can be used in

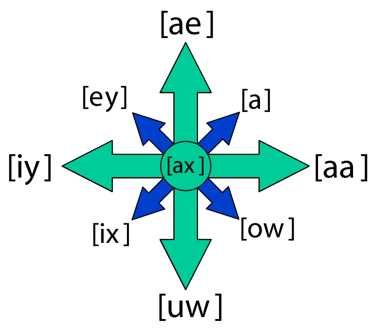


Figure 2: Vowel-direction mapping: vowels corresponding to directions.

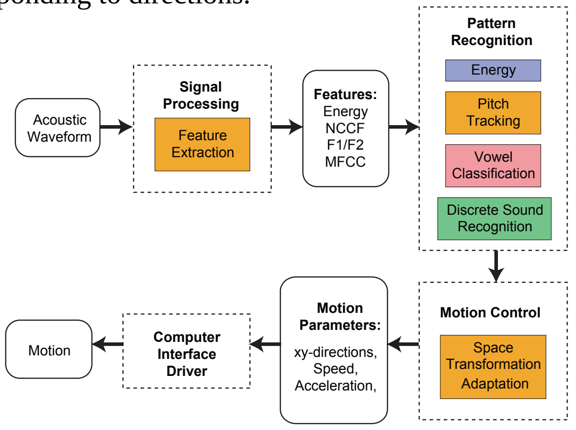

Acoustic Waveform
Signal Processing
Feature Extraction
Features: Energy NCCF F1/F2 MFCC
Pattern Recognition
Energy
Pitch Tracking
Vowel Classification
Discrete Sound Recognition
Motion Control
Space Transformation Adaptation
Motion Parameters: xy-directions, Speed, Acceleration,
Computer Interface Driver
Motion

Figure 3: System organization

estimating the vocal characteristics. The features we use are energy, normalized cross-correlation coefficients (NCCC), formant estimates, Mel-frequency cepstral coefficients (MFCCs), and formant estimates. To extract features, the speech signal is PCM sampled at a rate of F_s =16,000Hz. Energy is measured on a frame-by-frame basis with a frame size of 25ms and a frame step of 10ms. Pitch is extracted with a frame size of 40ms and a frame step of 10ms. Multiple pattern recognition tasks may share the same acoustic features: for example, energy and NCCCs are used for pitch tracking, and energy and MFCCs can be used in vowel classification and discrete sound recognition. Therefore, it is more efficient to decouple feature extraction from pattern recognition, as is shown in Figure 3.

3.2 Pattern Recognition

The pattern recognition module uses the acoustic features to extract desired parameters. The estimation and classification system must simultaneously perform *energy* computation (available from the input), *pitch tracking*, *vowel classification*, and *discrete sound recognition*.

Many state-of-the-art *pitch trackers* are based on dynamic programming (DP). This, however, often requires the meticulous design of local DP cost functions. The forms of these cost functions are usually empirically determined and/or their parameters are tuned by algorithms such as gradient descent (D.Talkin, 1995). Since different languages and applications may follow very different pitch transition patterns, the cost functions optimized for certain languages and applications may not be the most appropriate for others. Our VJ system utilizes a graphical model mechanism to automatically optimize the parameters of these cost functions, and has been shown to yield state-of-the-art performance (X.Li et al., 2004; J.Malkin et al., 2005).

For frame-by-frame *vowel classification*, our design constraints are the need for extremely low latency and low computational cost. Probability estimates for vowel classes thus need to be obtained as soon as possible after the vowel has been uttered or after any small change in voice quality has occurred. It is well known that models of vowel classification that incorporate temporal dynamics such as hidden Markov models (HMMs) can be quite accurate. However, the frame-by-frame latency requirements of VJ make HMMs unsuitable for vowel classification since HMMs estimate the likelihood of a model based on the entire utterance. An alternative is to utilize causal “HMM-filtering”, which computes likelihoods at every frame based on all frames seen so far. We have empirically found, however, that slightly non-causal and quite localized estimates of the vowel category probability is sufficient to achieve user satisfaction. Specifically, we obtain probability estimates of the form $p(V_t|X_{t-\tau},\ldots,X_{t+\tau})$, where V is a vowel class, and $X_{t-\tau},\ldots,X_{t+\tau}$ are feature frames within a length $2\tau+1$ window of features centered at time t. After several empirical trials, we decided on neural networks for vowel classification because of the availability of efficient discriminative training algorithms and their computational simplicity. Specifically we use a simple 2-layer multi-layer perceptron (Bishop, 1995) whose input layer consists of $26 * 7 = 182$ nodes, where 26 is the dimension of X_t, the MFCC feature vector, and $2\tau + 1 = 7$ is the number of consecutive frames, and that has 50 hidden nodes (the numbers 7 and 50 were determined empirically). The output layer has 4 output nodes representing 4 vowel probabilities. During training, the network is optimized to minimize the Kullback-Leibler (K-L) divergence between the output and the true label distribution, thus achieving the aforementioned probabilistic interpretation.

The VJ engine needs not only to detect that the user is specifying a vowel (for continuous control) but also a consonant-vowel-consonant (CVC) pattern (for discrete control) quickly and with a low probability of confusion (a VJ system also uses C and CV patterns for discrete commands). Requiring an initial consonant will phonetically distinguish these sounds from the pure vowel segments used for continuous control — the VJ system constantly monitors for changes that indicate the beginning of one of the discrete control commands. The vowel within the CV and CVC patterns, moreover, can help prevent background noise from being mis-classified as a discrete sound. Lastly, each such pattern currently requires an ending silence, so that the next command (a new discrete sound or continuous control vowel) can be accurately initiated. In all cases, a simple threshold-based rejection mechanism is used to reduce false positives.

To recognize the discrete control signals, HMMs are employed since, as in standard speech recognition, time warping is necessary to normalize for different signal durations corresponding to the same class. Specifically, we embed phone HMMs into "word" (C, CV, or CVC) HMMs. In this way, it is possible to train phone models using a training set that covers all possible phones, and then construct an application-specific discrete command vocabulary without retraining by recombining existing phone HMMs into new word HMMs. Therefore, each VJ-driven application can have its own appropriate discrete sound set.

3.3 Motion Control: Direction and Velocity

The VJ motion control module receives several pattern recognition parameters and processes them to produce output more appropriate for determining 2-D movement in the VJ-WIMP application.

Initial experiments suggested that using pitch to affect cursor velocity (Igarashi and Hughes, 2001) would be heavily constrained by an individual's vocal range. Giving priority to a more universal user-independent VJ system, we instead focused on relative energy. Our observation that users often became quiet when trying to move small amounts confirmed energy as a natural choice. Drastically different intrinsic average energy levels for each vowel, however, meant that comparing all sounds to a global average energy would create a large vowel-dependent bias. To overcome this, we distribute the energy per frame among the different vowels, in proportion to the probabilities output by the neural network, and track the average energy for each vowel independently. By splitting the power in this way, there is no effect when probabilities are close to 1, and we smooth out changes during vowel transitions when probabilities are more evenly distributed.

There are many possible options for determining velocity (a vector capturing both direction and speed magnitude) and "acceleration" (a function determining how the control-to-display ratio changes based on input parameters), and the different schemes have a large impact on user satisfaction. Unlike a standard mouse cursor, where the mapping is from 2-D hand movement to a 2-D screen, the VJ system maps from vocal-tract articulatory movement to a 2-D screen, and the transformation is not as straightforward. All values are for the current time frame t unless indicated otherwise. First, a raw direction value is calculated for each axis $j \in \{\text{x}, \text{y}\}$ as

$$d_j = \sum_i p_i \cdot \langle \mathbf{v}_i, \mathbf{e}_j \rangle \qquad (1)$$

in which $p_i = p(V_t = i | X_{t-\tau,\ldots,t+\tau})$ is the probability for vowel i at time t, $\mathbf{v}_i$ is a unit vector in the direction of vowel i, $\mathbf{e}_j$ is the unit-length positive directional basis vector along the j axis, and $\langle \mathbf{v}, \mathbf{e} \rangle$ is the projection of vector $\mathbf{v}$ onto unit vector $\mathbf{e}$. To determine movement speed, we first calculate a scalar for each axis j as

$$s_j = \sum_i \max \left[0, g_i \left(p_i \cdot f(\frac{E}{\mu_i}) \right) \right] \cdot |\langle \mathbf{v}_i, \mathbf{e}_j \rangle|$$

where E is the energy in the current frame, μ_i is the average energy for vowel i, and $f(\cdot)$ and $g_i(\cdot)$ are functions used for energy normalization and perceptual scaling (such as logs and/or cube-roots). This therefore allocates frame energy to direction based on the vowel probabilities. Lastly, we calculate the velocity for axis j at the current frame as

$$V_j = \beta \cdot s_j^{\alpha} \cdot \exp(\gamma s_j). \qquad (2)$$

where β represents the overall system sensitivity and the other values (α and γ) are warping constants, allowing the user to control the shape of the acceleration curve. Typically only one of α and γ is nonzero. Setting both to zero results in constant-speed movement along each axis, while $\alpha = 1$ and $\gamma = 0$ gives a linear mapping that will scale motion with energy but have no acceleration. The current user-independent system uses $\beta = 0.6$, $\gamma = 1.0$ and sets $\alpha = 0$. Lastly, the final velocity along axis j is $V_j d_j$. Future publications will report on systematic evaluations of different $f(\cdot)$ and $g_i(\cdot)$ functions.

3.4 Motion Control: User Adaptation

Since vowel quality is used for continuous control, inaccuracies can arise due to speaker variability owing to different speech loudness levels, vocal tract lengths, etc. Moreover, a vowel class articulated by one user might partially overlap in acoustic space with a different vowel class from another user. This imposes limitations on a purely user-independent vowel classifier. Differences in speaker loudness alone could cause significant unpredictability. To mitigate these problems, we have designed an adaptation procedure where each user is asked to pronounce four pre-defined vowel sounds, each lasting 2-3 seconds, at the beginning of a VJ session. We have investigated several novel adaptation strategies utilizing both neural networks and support vector machines (SVM). The fundamental idea behind them both is that an initial speaker-independent transformation of the space is learned using training data, and is represented by the first layer of a neural network. Adaptation data then is used to transform various parameters of the classifier (e.g., all or sub-portions of the neural network, or the parameters of the SVM). Further details of some of these novel adaptation strategies appear in (X.Li et al., 2005), and the remainder will appear in forthcoming publications. Also, the average energy values of each vowel for each user are recorded and used to normalize the speed control rate mentioned above. Preliminary evaluations on the data so far collected show very good results, with adaptation reducing the vowel classification error rate by 18% for the 4-class case, and 35% for the 8-class case. Moreover, informal studies have shown that users greatly prefer the VJ system after adaptation than before.

4 Applications and Videos

Our overall intent is for VJ to interface with a variety of applications, and our primary application so far has been to drive a standard WIMP interface with VJ controls, what we call the *VJ-WIMP* application. The current VJ version allows left button clicks (press and release, using the consonant [k]) as well as left button toggles (using consonant [ch]) to allow dragging. Since WIMP interfaces are so general, this allows us to indirectly control a plethora of different applications. Video demonstrations are available at the URL: `http://ssli.ee.washington.edu/vj`.

One of our key VJ applications is vocal web browsing. The video (dated 6/2005) shows examples of two web browsing tasks, one as an example of navigating the New York Times web site, the other using Google Maps to select and zoom in on a target area. Section 5 describes a preliminary evaluation on these tasks. We have also started using the VJ engine to control video games (third video example), have interfaced VJ with the Dasher system (Ward et al., 2000) (we call it the "Vocal Dasher"), and have also used VJ for figure drawing.

Several additional direct VJ-applications have also been developed. Specifically, we have directly interfaced the VJ system into a simple blocks world environment, where more precise object movement is possible than via the mouse driver. Specifically, this environment can draw arbitrary trajectories, and can precisely measure user fidelity when moving an object along a trajectory. Fidelity depends both on positional accuracy and task duration. One use of this environment shows the spatial direction corresponding to vocal effort (useful for training, forth video example). Another shows a simple robotic arm being controlled by VJ. We plan to use this environment to perform formal and precise user-performance studies in future work.

5 Preliminary User Study

We conducted a preliminary user study[1] to evaluate the feasibility of VJ and to obtain feedback regarding specific difficulties in using the VJ-WIMP system. While this study is not accurate in that: 1) it does not yet involve the intended target population

[1]The user study presented here used an earlier version of VJ than the current improved one described in the preceding pages.

of individuals with motor impairments, and: 2) the users had only a small amount of time to practice and become adept at using VJ, the study is still indicative of the VJ approach's overall viability as a novel voice-based human-computer interface method. The study quantitatively compares VJ performance with a standard desktop mouse, and provides qualitative measurement of the user's perception of the system.

5.1 Experiment Setup

We recruited seven participants ranging from age 22 to 26, none of whom had any motor impairment. Of the seven participants, two were female and five were male. All of them were graduate students in Computer Science, although none of them had previously heard of or used VJ. Four of the participants were native English speakers; the other three had an Asian language as their mother tongue.

We used a Dell Inspiron 9100 laptop with a 3.2 GHz Intel Pentium IV processor running the Fedora Core 2 operating system, with a 1280 x 800 24-bit color display. The laptop was equipped with an external Microsoft IntelliMouse connected through the USB port which was used for all of the tasks involving the mouse. A head-mounted Amanda NC-61 microphone was used as the audio input device, while the audio feedback from the laptop was output through the laptop speakers. The Firefox browser was used for all of the tasks, with the browser screen maximized such that the only portion of the screen which was not displaying the contents of the web page was the top navigation toolbar which was 30 pixels high.

5.2 Quantitative and Qualitative Evaluation

At the beginning of the quantitative evaluation, each participant was given a brief description of the VJ operations and was shown a demonstration of the system by a practiced experimenter. The participant was then guided through an adaptation process during which she/he was asked to pronounce the four directional vowels (Section 3.4). After adaptation, the participant was given several minutes to practice using a simple target clicking application. The quantitative portion of our evaluation followed a within-participant design. We exposed each participant to two experimental conditions which we refer to as input modalities: the *mouse* and the *VJ*. Each participant completed two tasks on each modality, with one trial per task.

The first task was a link navigation task (*Link*), in which the participants were asked to start from a specific web page and follow a particular set of links to reach a destination. Before the trial, the experimenter demonstrated the specified sequence of links to the participant by using the mouse and clicking at the appropriate links. The participant was also provided with a sheet of paper for their reference that listed the sequence of links that would lead them to the target. The web site we used was a Computer Science Department student guide and the task involved following six links with the space between each successive link including both horizontal and vertical components.

The second task was map navigation (*Map*), in which the participant was asked to navigate an online map application from a starting view (showing the entire USA) to get to a view showing a particular campus. The size of the map was 400x400 pixels, and the set of available navigation controls surrounding the map included ten discrete zoom level buttons, eight directional panning arrows, and a click inside the map causing the map to be centered and zoomed in by one level. Before the trial, a practiced experimenter demonstrated how to locate the campus map starting from the USA view to ensure they were familiar with the geography.

For each task, the participants performed one trial using the mouse, and one trial using a 4-class VJ. The trials were presented to the participants in a counterbalanced order. We recorded the completion time for each trial, as well as the number of *false positives* (system interprets a click when the user did not make a click sound), *missed recognitions* (the user makes a click sound but the system fails to recognize it as a click), and *user errors* (whenever the user clicks on an incorrect link). The recorded trial times include the time used by all of the above errors including recovery time.

After the completion of the quantitative evaluation, the participants were given a questionnaire which consisted of 14 questions related to the participants' perception of their experience using VJ such as the degree of satisfaction, frustration, and embarrassment. The answers were encoded on a 7-point Likert scale. We also included a space where the participants could write in any comments, and an in-

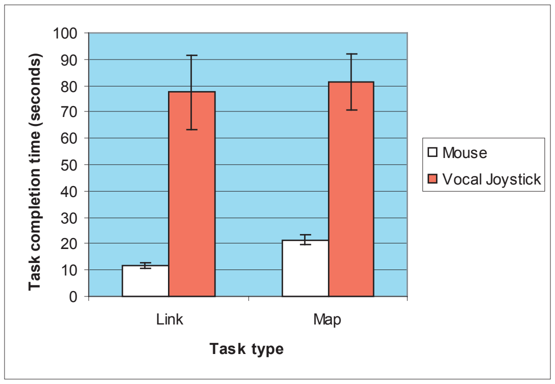


Figure 4: Task complement times

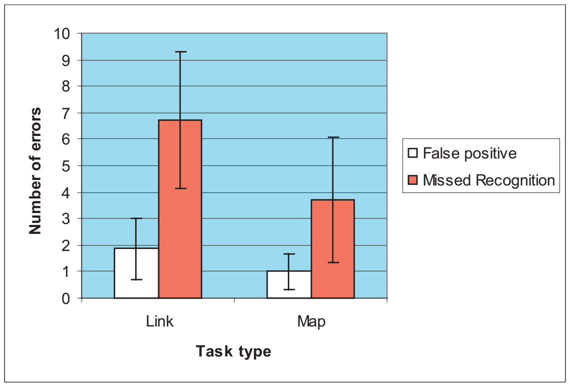


Figure 6: Average number of click errors per task

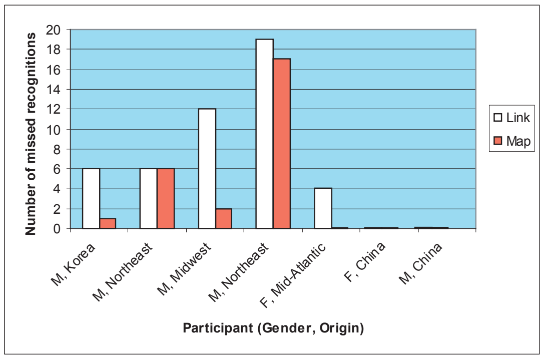


Figure 5: Missed recognitions by participant

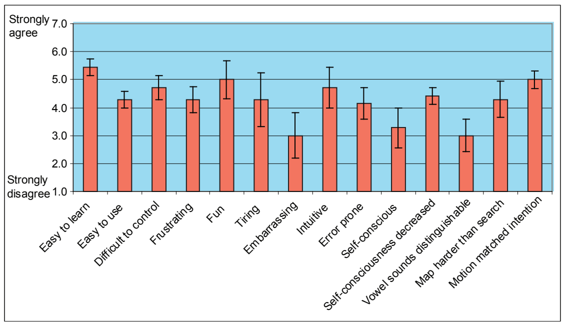


Figure 7: Questionnaire results

formal post-experiment interview was performed to solicit further feedback.

5.3 Results

Figure 4 shows the task completion times for Link and Map tasks, Figure 5 shows the breakdown of click errors by individual participants, Figure 6 shows the average number of false positive and missed recognition errors for each of the tasks. There was no instance of user error in any trial. Figure 7 shows the median of the responses to each of the fourteen questionnaire questions (error bars in each plot show ± standard error). In our measurement of the task completion times, we considered the VJ's recognition error rate as a fixed factor, and thus did not subtract the time spent during those errors from the task completion time.

There were several other interesting observations that were made throughout the study. We noticed that the participants who had the least trouble with missed recognitions for the clicking sound were either female or with an Asian language background, as shown in Figure 5. Our hypothesis regarding the better performance by female participants is that the original click sound was trained on one of our female researcher's voice. We plan also in future work to determine how the characteristics of different native language speakers influence VJ, and ultimately to correct for any bias.

All but one user explicitly expressed their confusion in distinguishing between the [ae] and [aa] vowels. Four of the seven participants independently stated that their performance would probably have been better if they had been able to practice longer, and did not attribute their perceived suboptimal performance to the quality of the VJ's recognition system. Several participants reported that they felt their vocal cords were strained due to having to produce a loud sound in order to get the cursor to move at the desired speed. We suspect this is due either to analog gain problems or to their adapted voice being too loud, and therefore the system calibrating the normal speed to correspond to the loud voice. We have since removed this problem by adjusting our adaptation strategy to express preference for a quiet voice.

In summary, the results from our study suggest that users without any prior experience were able to perform basic mouse based tasks using the Vocal Joystick system with relative slowdown of four to nine times compared to a conventional mouse. We anticipate that future planned improvements in the algorithms underlying the VJ engine (to improve accuracy, user-independence, adaptation, and speed) will further increase the VJ system's viability, and combined with practice could improve VJ enough so that it becomes a reasonable alternative compared to a standard mouse's performance.

6 Related Work

Related voice-based interface studies include (Igarashi and Hughes, 2001; Olwal and Feiner, 2005). Igarashi & Hughes presented a system where non-verbal voice features control a mouse system — their system requires a command-like discrete sound to determine direction before initiating a movement command, where pitch is used to control velocity. We have empirically found an energy-based mapping for velocity (as used in our VJ system) both more reliable (no pitch-tracking errors) and intuitive. Olwal & Feiner's system moves the mouse only after recognizing entire words. de Mauro's "voice mouse" `http://www.dii.unisi.it/~maggini/research/voice_mouse.html` focuses on continuous cursor movements similar to the VJ scenario; however, the voice mouse only starts moving after the vocalization has been completed leading to long latencies, and it is not easily portable to other applications. Lastly, the commercial dictation program Dragon by ScanSoft includes MouseGrid$^{\mathrm{TM}}$(Dra, 2004) which allows discrete vocal commands to recursively 9-partition the screen, thus achieving log-command access to a particular screen point. A VJ system, by contrast, uses continuous aspects of the voice, has change latency (about 60ms) not much greater than reaction time, and allows the user to make instantaneous directional change using one's voice (e.g., a user can draw a "U" shape in one breath).

7 Conclusions

We have presented new voice-based assistive technology for continuous control tasks and have demonstrated an initial system implementation of this concept. An initial user study using a group of individuals from the non-target population confirmed the feasibility of this technology. We plan next to further improve our system by evaluating a number of novel pattern classification techniques to increase accuracy and user-independence, and to introduce additional vocal characteristics (possibilities include vibrato, degree of nasality, rate of change of any of the above as an independent parameter) to increase the available simultaneous degrees of freedom controllable via the voice. Moreover, we plan to develop algorithms to decouple unintended user correlations of these parameters, and to further advance both our adaptation and acceleration algorithms.

References

C. Bishop. 1995. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

2004. Dragon naturally speaking, Mousegrid$^{\mathrm{TM}}$, ScanSoft Inc.

D.Talkin. 1995. A robust algorithm for pitch tracking (RAPT). In W.B.Kleign and K.K.Paliwal, editors, *Speech Coding and Synthesis*, pp. 495–515, Amsterdam. Elsevier Science.

X. Huang, A. Acero, and H.-W. Hon. 2001. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall.

T. Igarashi and J. F. Hughes. 2001. Voice as sound: Using non-verbal voice input for interactive control. In *ACM UIST 2001*, November.

J.Malkin, X.Li, and J.Bilmes. 2005. A graphical model for formant tracking. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

A. Olwal and S. Feiner. 2005. Interaction techniques using prosodic feature of speech and audio localization. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pp. 284–286.

D. Ward, A. F. Blackwell, and D. C. MacKay. 2000. Dasher - a data entry interface using continuous gestures and language models. In *ACM UIST 2000*.

X.Li, J.Malkin, and J.Bilmes. 2004. A graphical model approach to pitch tracking. In *Proc. Int. Conf. on Spoken Language Processing*.

X.Li, J.Bilmes, and J.Malkin. 2005. Maximum margin learning and adaptation of MLP classifers. In *9th European Conference on Speech Communication and Technology (Eurospeech'05)*, Lisbon, Portugal, September.

Speech-based Information Retrieval System with Clarification Dialogue Strategy

Teruhisa Misu **Tatsuya Kawahara**
School of informatics
Kyoto University
Sakyo-ku, Kyoto, Japan
misu@ar.media.kyoto-u.ac.jp

Abstract

This paper addresses a dialogue strategy to clarify and constrain the queries for speech-driven document retrieval systems. In spoken dialogue interfaces, users often make utterances before the query is completely generated in their mind; thus input queries are often vague or fragmental. As a result, usually many items are matched. We propose an efficient dialogue framework, where the system dynamically selects an optimal question based on information gain (IG), which represents reduction of matched items. A set of possible questions is prepared using various knowledge sources. As a bottom-up knowledge source, we extract a list of words that can take a number of objects and potentially causes ambiguity, using a dependency structure analysis of the document texts. This is complemented by top-down knowledge sources of metadata and hand-crafted questions. An experimental evaluation showed that the method significantly improved the success rate of retrieval, and all categories of the prepared questions contributed to the improvement.

1 Introduction

The target of spoken dialogue systems is being extended from simple databases such as flight information (Levin et al., 2000; Potamianos et al., 2000) to general documents (Fujii and Itou, 2003) including newspaper articles (Chang et al., 2002; Hori et al., 2003). In such systems, the automatic speech recognition (ASR) result of the user utterance is matched against a set of target documents using the vector space model, and documents with high matching scores are presented to the user.

In this kind of document retrieval systems, user queries must include sufficient information to identify the desired documents. In conventional document query tasks with typed-text input, such as TREC QA Track (NIST and DARPA, 2003), queries are (supposed to be) definite and specific. However, this is not the case when speech input is adopted. The speech interface makes input easier. However, this also means that users can start utterances before queries are thoroughly formed in their mind. Therefore, input queries are often vague or fragmental, and sentences may be ill-formed or ungrammatical. Moreover, important information may be lost due to ASR errors. In such cases, an enormous list of possible relevant documents is usually obtained because there is very limited information that can be used as clues for retrieval. Therefore, it is necessary to narrow down the documents by clarifying the user's intention through a dialogue.

There have been several studies on the follow-up dialogue, and most of these studies assume that the target knowledge base has a well-defined structure. For example, Denecke (Denecke and Waibel, 1997) addressed a method to generate guiding questions based on a tree structure constructed by unifying pre-defined keywords and semantic slots. However, these approaches are not applicable to general docu-

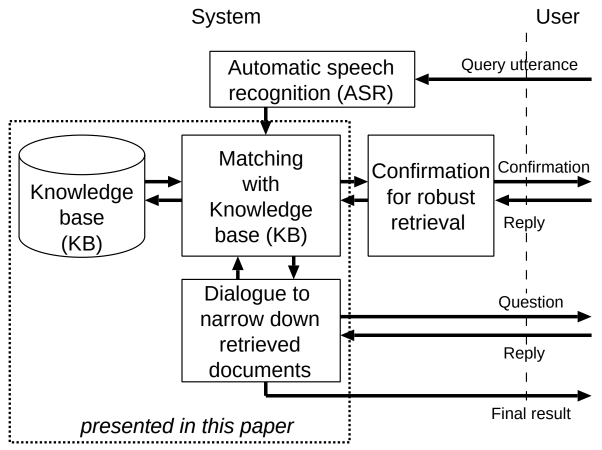

System
User
Automatic speech recognition (ASR)
Query utterance
Knowledge base (KB)
Matching with Knowledge base (KB)
Confirmation for robust retrieval
Confirmation
Reply
Dialogue to narrow down retrieved documents
Question
Reply
Final result
presented in this paper

Figure 1: System overview

ment sets without such structures.

In this paper, we propose a dialogue strategy to clarify the user's query and constrain the retrieval for a large-scale text knowledge base, which does not have a structure nor any semantic slots. In the proposed scheme, the system dynamically selects an optimal question, which can reduce the number of matched items most efficiently. As a criterion of efficiency of the questions, information gain (IG) is defined. A set of possible questions is prepared using bottom-up and top-down knowledge sources. As a bottom-up knowledge source, we conduct dependency structure analysis of the document texts, and extract a list of words that can take a number of objects, thus potentially causing ambiguity. This is combined with top-down knowledge sources of metadata and hand-crafted questions. The system then updates the query sentence using the user's reply to the question, so as to generate a confirmation to the user.

2 Document retrieval system for large-scale knowledge base

2.1 System overview

We have studied a dialogue framework to overcome the problems in speech-based document retrieval systems. In the framework, the system can handle three types of problems caused by speech input: ASR errors, redundancy in spoken language expression, and vagueness of queries. First, the system realizes robust retrieval against ASR errors and redundancies by detecting and confirming them. Then, the system makes questions to clarify the user's query and narrow down the retrieved documents.

The system flow of these processes is summarized below and also shown in Figure 1.

1. Recognize the user's query utterance.
2. Make confirmation for phrases which may include critical ASR errors.
3. Retrieve from knowledge base (KB).
4. Ask possible questions to the user and narrow down the matched documents.
5. Output the retrieval results.

In this paper, we focus on the latter stage of the proposed framework, and present a clarification dialogue strategy to narrow down documents.

Table 1: Document set (Knowledge Base: KB)

Text collection	# documents	text size (byte)
glossary	4,707	1.4M
FAQ	11,306	12M
DB of support articles	23,323	44M

2.2 Task and back-end retrieval system

Our task involves text retrieval from a large-scale knowledge base. For the target domain, we adopt a software support knowledge base (KB) provided by Microsoft Corporation. The knowledge base consists of the following three kinds: glossary, frequently asked questions (FAQ), and support articles. The specification is listed in Table 1, and there are about 40K documents in total. An example of support article is shown in Figure 2.

Dialog Navigator (Kiyota et al., 2002) has been developed at University of Tokyo as a retrieval system for this KB. The system accepts a typed-text input from users and outputs a result of the retrieval. The system interprets an input sentence by taking syntactic dependency and synonymous expression into consideration for matching it with the KB. The target of the matching is the summaries and detail information in the support articles, and the titles of the Glossary and FAQ. The retrieved result is displayed to the user as the list of documents like Web

HOWTO:
Use Speech Recognition in Windows XP
The information in this article applies to:

- Microsoft Windows XP Professional
- Microsoft Windows XP Home Edition

Summary: This article describes how to use speech recognition in Windows XP. If you installed speech recognition with Microsoft Office XP, or if you purchased a new computer that has Office XP installed, you can use speech recognition in all Office programs as well as other programs for which it is enabled.

Detail information: Speech recognition enables the operating system to convert spoken words to written text. An internal driver, called a speech recognition engine, recognizes words and converts them to text. The speech recognition engine ...

Figure 2: Example of software support article

search engines. Since the user has to read detail information of the retrieved documents by clicking their icons one by one, the number of items in the final result is restricted to about 15.

In this work, we adopt Dialog Navigator as a back-end system and construct a spoken dialogue interface.

3 Dialogue strategy to clarify user's vague queries

3.1 Dialogue strategy based on information gain (IG)

In the proposed clarification dialogue strategy, the system asks optimal questions to constrain the given retrieval results and help users find the intended ones. Questions are dynamically generated by selecting from a pool of possible candidates that satisfy the precondition. The information gain (IG) is defined as a criterion for the selection. The IG represents a reduction of entropy, or how many retrieved documents can be eliminated by incorporating additional information (a reply to a question in this case). Its computation is straightforward if the question classifies the document set in a completely disjointed manner. However, the retrieved documents may belong to two or more categories for some questions, or may not belong to any category. For example, some documents in our KB are related with multiple versions of MS-Office, but others may be irrelevant to any of them. Moreover, the matching score of the retrieved documents should be taken into account in this computation. Therefore, we define IG $H(S)$ for a candidate question S by the following equations.

$$H(S) = -\sum_{i=0}^{n} P(i) \cdot \log P(i)$$

$$P(i) = \frac{|C_i|}{\sum_{i=0}^{n} |C_i|}$$

$$|C_i| = \sum_{D_k \in i} CM(D_k)$$

Here, D_k denotes the k-th retrieved document by matching the query to the KB, and $CM(D)$ denotes the matching score of document D. Thus, C_i represents the number of documents classified into category i by candidate question S, which is weighted with the matching score. The documents that are not related to any category are classified as category 0.

The system flow incorporating this strategy is summarized below and also shown in Figure 3.

1. For a query sentence, retrieve from KB.
2. Calculate IG for all possible candidate questions which satisfy precondition.
3. Select the question with the largest IG (larger than a threshold), and ask the question to the user. Otherwise, output the current retrieval result.
4. Update the query sentence using the user's reply to the question.
5. Return to 1.

This procedure is explained in detail in the following sections.

3.2 Question generation based on bottom-up and top-down knowledge sources

We prepare a pool of questions using three methods based on bottom-up knowledge together with top-down knowledge of KB. For a bottom-up knowledge

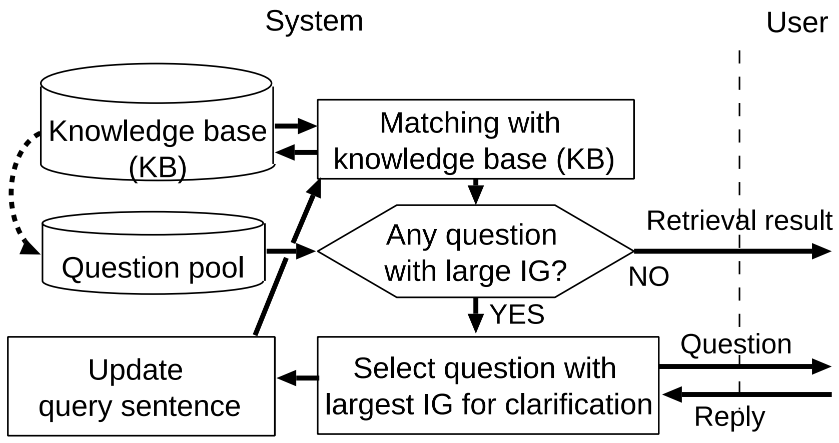

System
User
Knowledge base (KB)
Matching with knowledge base (KB)
Question pool
Any question with large IG?
Retrieval result
NO
YES
Question
Reply
Update query sentence
Select question with largest IG for clarification

Figure 3: Overview of query clarification

Table 2: Examples of candidate questions (Dependency structure analysis: method 1)

Question	Precondition	Ratio of applicable doc.	IG
What did you <u>delete</u>?	Query sentence includes “delete”	2.15 (%)	7.44
What did you <u>install</u>?	Query sentence includes “install”	3.17 (%)	6.00
What did you <u>insert</u>?	Query sentence includes “insert”	1.12 (%)	7.12
What did you <u>save</u>?	Query sentence includes “save”	1.81 (%)	6.89
What is the <u>file</u> type?	Query sentence includes “file”	0.94 (%)	6.00
What did you <u>setup</u>?	Query sentence includes “setup”	0.69 (%)	6.45

source, we conducted a dependency structure analysis on KB. As for top-down knowledge, we make use of metadata included in KB and human knowledge.

3.2.1 Questions based on dependency structure analysis (method 1)

This type of question is intended to clarify the modifier or object of some words, based on dependency structure analysis, when they are uncertain. For instance, the verb “delete” can have various objects such as “application program” or “address book”. Therefore, the query can be clarified by identifying such objects if they are missing. However, not all words need to be confirmed because the modifier or object can be identified almost uniquely for some words. For instance, the object of the word “shutdown” is “computer” in most cases in this task domain. It is tedious to identify the object of such words. We therefore determine the words to be confirmed by calculating entropy for modifier-head pairs from the text corpus. The procedure is as follows.

1. Extract all modifier-head pairs from the text of KB and query sentences (typed input) to another retrieval system[1] provided by Microsoft Japan.
2. Calculate entropy $H(m)$ for every word based on probability $P(i)$. This $P(i)$ is calculated with the occurrence count $N(m)$ of word m that appears in the text corpus and the count $N(i, m)$ of word m whose modifier is i.

$$H(m) = -\sum_i P(i) * \log P(i)$$

$$P(i) = \frac{N(i, m)}{N(m)}$$

[1]http://www.microsoft.com/japan/enable/nlsearch/

Table 3: Examples of candidate questions (Metadata: method 2)

Question	Precondition	Ratio of applicable doc.	IG
What is the version of your Windows?	None	30.03 (%)	2.63
What is your application?	None	30.28 (%)	2.31
What is the version of your Word?	Query sentence includes “Word”	3.76 (%)	2.71
What is the version of your Excel?	Query sentence includes “Excel”	4.13 (%)	2.44

Table 4: List of candidate questions (Human knowledge: method 3)

Question	Precondition	Ratio of applicable doc.	IG
When did the symptom occur?	None	15.40 (%)	8.08
Tell me the error message.	Query sentence includes “error”	2.63 (%)	8.61
What do you concretely want to do?	None	6.98 (%)	8.04

As a result, we selected 40 words that have a large value of entropy. Question sentences for these words were generated with a template of “What did you ...?” and unnatural ones were corrected manually. Categories for IG calculation are defined by objects of these words included in matched documents. The system can make question using this method when these words are included in the user’s query. Table 2 lists examples of candidate questions using this method. In this table, ratio of applicable document corresponds to the ratio of documents that include the words selected above, and IG is calculated using applicable documents.

3.2.2 Questions based on metadata included in KB (method 2)

We also prepare candidate questions using the metadata attached to the KB. In general large-scale KBs, metadata is usually attached to manage them efficiently. For example, category information is attached to newspaper articles and books in libraries. In our target KB, a number of documents include metadata of product names to which the document applies. The system can generate question to which the user’s query corresponds using this metadata. However, some documents are related with multiple versions, or may not belong to any category. Therefore, the performance of these questions greatly depends on the characteristics of the metadata.

Fourteen candidate questions are prepared using this method. Example of candidate questions are listed in Table 3. Ratio of applicable document corresponds to the ratio of documents that have metadata of target products.

3.2.3 Questions based on human knowledge (method 3)

Software support is conventionally provided by operators at call centers. We therefore prepare candidate questions based on the human knowledge that has been accumulated there. This time, three kinds of questions are hand-crafted. For instance, the question “When did the symptom occur?” tries to capture key information to identify relevant documents. The categories for IG caluclation are defined using hand-crafted rules by focusing on key-phrases such as “after ...” or “during ...”. Candidate questions are listed in Table 4.

An example dialogue where the system asks questions based on IG is in Figure 4.

3.3 Update of retrieval query sentence

Through the dialogue to clarify the user’s query, the system updates the query sentence using the user’s reply to the question. Our backend information retrieval system does not adopt simple “bag-

S1:	What is your problem?
U1:	Too garbled to read.
(Retrieval results):	1. Close button and maximize button are garbled. 2. Characters are garbled in Outlook Today. 3. Characters are garbled while inserting Japanese text. 4. VB application is garbled to read. . . .
	(Calculate IG) · Candidate question 1: What is garbled to read? – IG 5.27 · Candidate question 2: What is the version of your Windows? – IG 1.43 · Candidate question 3: When did the symptom occur? – IG 2.47 . . .
S2:	**(Select question with largest IG)** What is garbled to read?
U2:	Characters on window button.
S3:	**(Update query sentence)** Retrieving with "Characters on window button are too garbled to read".

Figure 4: Example dialogue

of-words" model, but conducts a more precise dependency structure analysis for matching; therefore forming an appropriate query sentence is desirable rather than simply adding keywords. Moreover, it is more comprehensible to the user to present the updated query sentence than to show the sequence of ASR results. Here, the update rules of the query sentence are prepared as follows.

1. Questions based on dependency structure analysis
   The user's reply is added immediately before of after the word that is the reply's modifying head. For instance, the reply to the question "What did you delete?" is inserted right after the word "delete" in the query sentence.

2. Questions based on metadata of KB
   Phrases "In {Product name} {version name}" are added to the query sentence.

3. Questions based on human knowledge
   The position where the user's reply is inserted is specified beforehand for each question candidate. For instance, the reply to the question "Tell me the error message." is inserted right after the word "error" in the query sentence.

A dialogue example where the system updates the user's query is shown in Figure 5. In the example, the system makes confirmation "Retrieving with 'When I try to open it in explorer, I cannot open Excel 2002 file' " at the end of the dialogue before presenting the actual retrieval result.

3.4 Experimental evaluation

We implemented and evaluated the proposed method. We collected a test data by 14 subjects who had not used our system Each subject was requested to retrieve support articles for 14 tasks, which consisted of prepared scenarios (query sentences were not given). The subjects were allowed to utter a query again up to twice per task if they thought an adequate retrieval result was not obtained. As a result, we collected 238 utterances for 196 (=14 $\times$ 14) tasks in total. An example of scenario and user utterances are shown in Figure 6. The average word accuracy of ASR was 82.9%. The threshold value in IG that the system makes a question is set to 1.0 ini-

S1:	What is your problem?
U1:	I cannot open the file.
S2:	What is the file type? (method 1)
U2:	Excel file.
(Update query sentence):	“I cannot open Excel file.”
S3:	What is the version of your Excel? (method 2)
U3:	My Excel is version 2002.
(Update query sentence):	“I cannot open Excel 2002 file.”
S4:	When did the symptom occur? (method 3)
U4:	Tried to open it in explorer.
S5:	Retrieving with “When I try to open it in explorer, I cannot open Excel 2002 file”.

Figure 5: Query sentence update using user’s reply

- **An example of scenario**
  You are looking for restaurant in Kyoto using WWW. You have found a nice restaurant and tried to print out an image of the map showing the restaurant. However, it is not printed out. (Your browser is IE 6.0)
- **Examples of users’ utterance**
  - I want to print an image of map.
  - I can’t print out.
  - I failed to print a picture in homepage using IE.
  - Please tell me how to print out an image.

Figure 6: Example of scenario and user utterances

tially, and incremented by 0.3 every time the system generates a question through a dialogue session.

First, we evaluated the success rate of retrieval. We regarded a retrieval as successful when the retrieval result contained a correct document entry for the scenario. We compared the following cases.

1. Transcript: A correct transcript of the user utterance, prepared manually, was used as an input.

2. ASR result (baseline): The ASR result was used as an input.

3. Proposed method (log data): The system generated questions based on the proposed method, and the user replied to them as he/she thought appropriate.

We also evaluated the proposed method by simulation in order to confirm its theoretical effect. Various factors of the entire system might influence the performance in real dialogue which is evaluated by the log data. Specifically, the users might not have answered the questions appropriately, or the replies might not have been correctly recognized. Therefore, we also evaluated with the following condition.

4. Proposed method (simulation): The system generated questions based on the proposed method, and appropriate answers were given manually.

Table 5 lists the retrieval success rate and the rank of the correct document in the retrieval result, by these cases. The proposed method achieved a better success rate than when the ASR result was used. An improvement of 12.6% was achieved in the simulation case, and 7.7% by the log data. These figures demonstrate the effectiveness of the proposed approach. The success rate of the retrieval was about 5% higher in the simulation case than the log data. This difference is considered to be caused by following factors.

1. ASR errors in user’s uttered replies
   In the proposed strategy, the retrieval sentence is updated using the user’s reply to the question regardless of ASR errors. Even when the user notices the ASR errors, he/she cannot correct them. Although it is possible to confirm them using ASR confidence measures, it makes dialogue more complicated. Hence, it was not implemented this time.

2. User’s misunderstanding of the system’s questions
   Users sometimes misunderstood the system’s questions. For instance, to the system question “When did the symptom occur?”, some user

Table 5: Success rate and average rank of correct document in retrieval

	Success rate	Rank of correct doc.
Transcript	76.1%	7.20
ASR result (baseline)	70.7%	7.45
Proposed method (log data)	78.4%	4.40
Proposed method (simulation)	83.3%	3.85

Table 6: Comparison of question methods

	Success rate	# generated questions (per dialogue)
ASR result (baseline)	70.7%	—
Dependency structure analysis (method 1)	74.5%	0.38
Metadata (method 2)	75.7%	0.89
Human knowledge (method 3)	74.5%	0.97
All methods (method 1-3)	83.3%	2.24

replied simply “just now” instead of key information for the retrieval. To this problem, it may be necessary to make more specific questions or to display reply examples.

We also evaluated the efficiency of the individual methods. In this experiment, each of the three methods was used to generate questions. The results are in Table 6. The improvement rate by the three methods did not differ very much, and most significant improvement was obtained by using the three methods together. While the questions based on human knowledge are rather general and were used more often, the questions based on the dependency structure analysis are specific, and thus more effective when applicable. Hence, the questions based on the dependency structure analysis (method 1) obtained a relatively high improvement rate per question.

4 Conclusion

We proposed a dialogue strategy to clarify user' queries for document retrieval tasks. Candidate questions are prepared based on the dependency structure analysis of the KB together with KB metadata and human knowledge. The system selects an optimal question based on information gain (IG). Then, the query sentence is updated using the user's reply. An experimental evaluation showed that the proposed method significantly improved the success rate of retrieval, and all categories of the prepared questions contributed to the improvement.

The proposed approach is intended for restricted domains, where all KB documents and several knowledge sources are available, and it is not applicable to open-domain information retrieval such as Web search. We believe, however, that there are many targets of information retrieval in restricted domains, for example, manuals of electric appliances and medical documents for expert systems. The methodology proposed here is not so dependent on the domains, thus applicable to many other tasks of this category.

5 Acknowledgements

The authors are grateful to Prof. Kurohashi and Dr. Kiyota at University of Tokyo and Dr. Komatani at Kyoto University for their helpful advice.

References

E. Chang, F. Seide, H. M. Meng, Z. Chen, Y. Shi, and Y. C. Li. 2002. A system for spoken query information retrieval on mobile devices. *IEEE Trans. on Speech and Audio Processing*, 10(8):531–541.

M. Denecke and A. Waibel. 1997. Dialogue strategies guiding users to their communicative goals. In *Proc. EUROSPEECH*.

A. Fujii and K. Itou. 2003. Building a test collection for speech-driven Web retrieval. In *Proc. EUROSPEECH*.

C. Hori, T. Hori, H. Isozaki, E. Maeda, S. Katagiri, and S. Furui. 2003. Deriving disambiguous queries in a spoken interactive ODQA system. In *Proc. IEEE-ICASSP*.

Y. Kiyota, S. Kurohashi, and F. Kido. 2002. ”Dialog Navigator”: A question answering system based on large text knowledge base. In *Proc. COLING*, pages 460–466.

E. Levin, S. Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Di Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. 2000. The AT&T-DARPA Communicator mixed-initiative spoken dialogue system. In *Proc. ICSLP*.

NIST and DARPA. 2003. The twelfth Text REtrieval Conference (TREC 2003). In *NIST Special Publication SP 500–255*.

A. Potamianos, E. Ammicht, and H.-K. J. Kuo. 2000. Dialogue management in the Bell labs Communicator system. In *Proc. ICSLP*.

Learning Mixed Initiative Dialog Strategies By Using Reinforcement Learning On Both Conversants

Michael S. English and **Peter A. Heeman**
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
Beaverton OR, 97006, USA
menglish6@gmail.com and heeman@cslu.ogi.edu

Abstract

This paper describes an application of reinforcement learning to determine a dialog policy for a complex collaborative task where policies for both the system and a proxy for a user of the system are learned simultaneously. With this approach a useful dialog policy is learned without the drawbacks of other approaches that require significant human interaction. The specific task that the agents were trained on was chosen for its complexity and requirement that both conversants bring task knowledge to the interaction, thus ensuring its collaborative nature. The results of our experiment show that you can use reinforcement learning to create an effective dialog policy, which employs a mixed initiative strategy, without the drawbacks of large amounts of data or significant human input.

1 Introduction

The problem of developing a dialog manager can be expressed as the task of building a specific dialog policy for the dialog system to follow as it interacts with the user. A dialog policy can be thought of as an enumeration of all of the states a dialog system can be in, and the corresponding action to take from each of those states. Thus a policy completely specifies the behavior of a dialog manager.

Most conventional approaches to accomplishing this task seek to directly model human interactions in some manner. These techniques include hand-crafting a policy, using a Wizard-of-Oz approach in an iterative manner and inducing a policy from a human-human dialog corpus. All three approaches have shortcomings that make them less than ideal for developing dialog systems. The approach of hand-crafting of a dialog policy is problematic as it is difficult to predict how a user with interact with it, making it difficult to craft an optimal policy. To get around this, an iterative approach can be used, with a Wizard taking the place of the system. However, it is still difficult to train a wizard, and it is difficult to explore many different strategies in order to find the optimal one. Human-human dialog can be used for policy generation, as this should represent optimal behavior to accomplish a task. However, computers are not capable of behaving exactly as a human. In addition, humans might not interact with a computer as they would another person.

Recently a number of researchers have proposed using reinforcement learning to alleviating the problems encountered with more conventional methods of developing dialog policies. With the development of a good policy evaluation function, reinforcement learning can effectively and quickly explore a large policy space. There is the additional benefit that it will learn a policy that is optimal for the capabilities of the system.

The main drawback of reinforcement learning approaches is that they require some form of conversational partner to train the system against. Conventionally, these partners have taken the form of a human (Walker, 2000; Singh et al., 2002) or a simulated user (Levin et al., 2000; Scheffler and Young, 2002; Georgila et al., 2005). These two types of conversational partners limit the complexity and diversity of policies that can be generated by reinforcement learning. These two approaches to training partners limit the whole system to the abilities of the partners themselves. For a human partner we run into the significant time and effort problems that were present in Wizard-of-Oz and handcrafting policy development. With a simulated user the system is limited by the complexity and flexibility of the simulated user, which itself can require a large degree of handcrafting by its creator.

In this paper, we propose a solution to the conversational partner problem of generating a dialog policy with reinforcement learning. We have taken a complex collaborative task and used reinforcement learning, applied to both participants, to develop a dialog policy for the task. By training both agents simultaneously we are able to avoid the uncertainties of creating a user to train against, as well as the time and data limitations of training directly against humans. Our training approach allows us to avoid these conventional drawbacks even while applying reinforcement learning to complex tasks.

Section 2 provides a brief overview of previous work in using reinforcement learning for dialog systems. Sections 3 and 4 describe the dialog task and its specification as a reinforcement-learning problem. Section 5 and 6 present the results of this experiment and a discussion of them.

2 Related Work

A number of researchers have explored using reinforcement learning to create a policy for a dialog system. Walker (2000) trained a dialog system, ELVIS, to learn a dialog strategy for supporting spoken language access to a user's email. The main function of ELVIS is to provide verbal summaries of email folders. This summary could consist of simple statements about the number of messages or a more detailed description of current emails.

Reinforcement learning is used to determine the best settings for a variety of properties of the system. For example, the system must learn to choose between email reading styles of reading back the full email first, reading a summary of the email first, or prompting the user with the two choices of reading styles. The system also learns whether it is better to take a mixed initiative or a system initiative strategy when interacting with the user.

To enable the learning process, ELVIS utilized human users as its conversational partner. Users performed a set of tasks with ELVIS, with each run using different state-property values, which were randomly chosen for that dialog. In order to support humans as a training partner Walker restricted the policy space so that it would only contain policies that were capable of accomplishing the available system tasks. Thus, during training the users would not be faced with a system that simply could not perform the tasks asked of it.

ELVIS was trained with a Q-learning approach that sought to determine the expected utility at each state, where utility was a subjective function involving such variables as task completion and user satisfaction. The state variables utilized in the training process were (a) whether the user's name is known, (b) what the initiative style is, (c) the task progress, and (d) what the user's current goal is. Given these state variables, ELVIS was able to learn the best style to adopt in responding to the user's requests at various points in the dialog. One major shortcoming of the conversational partner used with ELVIS is its reliance upon human interaction for training. This shortcoming is somewhat mitigated by the fact that the learning problem was one of fitting together preexisting policy components, but would be severely limiting if the goal was to learn a complete dialog policy. The amount of data necessary for learning a complete policy makes direct human interaction in the learning process unrealistic.

Levin et al. (2000) tackles a slightly different reinforcement-learning task. She is learning a policy to use in a dialog system built from a small set of atomic actions. This system is trained to provide a verbal interface to an airline flight database. This system is able to provide users with a way to find flights that meet a dynamic set of criteria. The dialog agent's state consists of information regarding the departure city, destination city, flight date, etc. Levin takes a useful approach in reducing the size of true state space by simply tracking when a particular state variable has a value rather than including the specific value in the state. For instance during a dialog when the system determines that the departure city is New York it does not distinguish this from when it has determined that the departure city is Chicago.

To converse with the dialog agent during reinforcement learning, Levin uses a "simulated user." The simulated user is created from a corpus of human dialogs with a prior airline system. In developing this user Levin makes the simplifying assumption that a user's response is based solely on the previous prompt. Then the specific probabilities for each user response are determined by examining the corpus for exchanges that match the possible prompts for the new dialog agent as well as hand crafting some of the probabilities. During the actual learning the agent used Monte Carlo training with exploring starts in order to fully explore the state space.

The "simulated user" method of supplying the conversational partner seems difficult and not particularly applicable to tasks where a dialog corpus does not already exist, but Kearns and Singh (1998) indicates that the accuracy of the transition probabilities for the probabilistic user is not critical for the dialog agent to learn an optimal strategy. While this experiment does allow for the dialog agent to learn a complex strategy, the notion of learning against a simulated user limits the space of policies that will be considered during training. Training against a conversational partner that is a model of a human automatically prejudices the system towards policies that we would be inclined towards building by hand and precludes the sincere exploration of all possible policies.

3 Task Specification

For our experiment we use the task presented in Yang and Heeman (2004), which is a modification of the DesignWorld task of Walker (1995). The task requires 2 conversants to agree on 5 pieces of furniture to place in a room. Both conversants know all of the furniture items that can be chosen, which differ by color, type and point value. Each conversant also has private preferences about which furniture items it wants in the room; such as 'if there is a red couch in the room, I also want a lamp in the room'. Each preference has a score. As this is a collaborative task, the conversants have the goal of finding the 5 furniture items that have the highest score, where the score is the sum of the point value of each of the 5 chosen furniture items less the scores for any violated preferences of either conversant.

The conversational agents work to achieve their goal by performing the following actions: **propose**, **accept**, **reject**, **inform**, and **release turn**. If there is not a current proposal, either agent can **propose** an item, which makes that item into the current proposal. If there is a current proposal, the other conversant can **accept** it or **reject** it. Accepting an item results in that item being included in the task solution and removes it as the current proposal. Rejecting a proposed item removes it as the current proposal. When an item has been rejected it remains a valid choice for future proposals. In addition to accepting or rejecting a proposal, either conversant may **inform** the other conversant of preferences that are violated by the current proposal. A preference is violated by the current proposal if the addition of that proposed item to the solution set would cause the solution set to violate the preference. When a conversant informs of a violated preference, that preference becomes mutually known and so affects future decisions by both participants. Only preferences that are not known by the other conversant are communicated. For turn taking, we include the action **release turn**, which the conversant that currently has the turn can perform to signal that it is relinquishing the turn (cf. Traum and Hinkelman, 1992). Note that after a release turn, the other agent must make the next move, which could itself be a release turn. The inclusion of this action allows conversants to perform multiple actions in a row, such as a reject, an inform, and a propose. Our approach to turn taking differs slightly from Yang and Heeman, as they make it an implicit part of other actions.

In order to successfully utilize these actions in a dialog, some reasoning effort is required of the conversants. Conversants must be able to determine what preferences are violated by a pending proposal and which of the remaining items makes the best proposal. In order to keep the reasoning effort manageable, we follow Yang and Heeman and use a greedy algorithm to pick the item that results in the best score for the item plus the set of items already accepted. The conversants do not consider interactions with the items that will be subsequently added to the plan. Conversants using this greedy approach can construct a plan that is very close to optimal.

4 Learning Specification

4.1 Agent Specification

In order to apply reinforcement learning to this task we must formalize the conversants as reinforcement learning agents, specifying their state and actions, as well as the environment they will interact in. In order to reduce the size of the state space for this task we simplified the representation of the state in a manner similar to that done by Levin (2004). We formulated the state of the dialog agents with many of the more specific details of the actual state of the task removed. For instance the agent state does not include specific information about the furniture item that is the pending proposal, rather the agent's state only indicates that there is a pending proposal.

The state specification for each agent includes the following binary variables: **Pending-Proposal**, **I-Proposed**, **Violated-Preference**, **Prior-Violated-Preferences**, and **Better-Alternative**. **Pending-Proposal** indicates whether an item has been proposed but not accepted or rejected. **I-Proposed** indicates if the agent made the most recent proposal. **Violated-Preference** indicates that the pending proposal has caused one or more violations of the conversant's private preferences. **Prior-Violated-Preferences** indicates whether the conversant had one or more violated preferences when the pending proposal was made. This variable allows the agent to remember what its original response to a proposal was, even after it may have shared all of its preferences that were violated (thus creating a state where it no longer has any violated personal preferences). **Better-Alternative** indicates that the agent thinks it knows an item that would achieve a better score than the item currently proposed.

The actions from Section 3 can be sequenced in a number of different orders, leading to different policies. Unlike Yang and Heeman, who compared handcrafted policies, we use reinforcement learning to learn policy pairs, one part of the pair for the system, and the other for the simulated user. We have restricted the space of policies that can be learned. First, we reduce the space by only considering legal sequences of actions. For example, if there is a pending proposal, another item cannot be proposed. Second, after 5 items have been accepted, the dialog is automatically ended. Third, to keep the space of dialog policies small, we force an inform to inform of all violated preferences at once.

The Reinforcement Learning states and actions of our dialog agents capture a subset of the true state of the dialog. Our agents do not have the ability to distinguish between, or develop distinct policies in response to, the proposal of a blue chair versus a red desk. Since our formulation of the dialog agents do not encode specific information about items or preferences, the dialog environment must maintain these details. This extra information that must include the currently proposed item, what each agent's private and currently violated preferences are, what preferences are shared between each agent, what items have been accepted as part of the task solution, and what items are still available for selection. This technique of generalizing the state space is the same as the one used by Levin (2000), and allows us to keep the state space at a manageable size for our task.

4.2 Reinforcement Learning

For our Reinforcement Learning algorithm we chose to use an on-policy Monte Carlo method (Sutton and Barto, 1998). Our chosen task is naturally episodic since the two agents agreeing upon five items indicates task completion and thus the end of the dialog, which constitutes one learning episode. We also imposed a limit of 500 interactions per dialog in order to ensure that each learning episode was finite even if the task was not successfully completed. For some state-action pairs our task does not allow the accurate specification of the resulting state. In fact, due to the way that our state representation simplifies the true task environment an action choice for many states will necessarily lead to different states depending upon the task environment. For instance, proposing an item will sometimes lead to that items acceptance and sometimes it will be rejected. Given this uncertainty our learning approach necessarily had to learn the expected rewards of actions instead of states.

At the end of each dialog the interaction is given a score based on the evaluation function and that score is used to update the dialog policy of both agents. The state-action history for each agent is iterated over separately and the score from the recent dialog is averaged in with the expected return from the existing policy. We chose not to include any discounting factor to the dialog score as we progressed back through the dialog history. The decision to equally weight each state-action pair in the dialog history was made because an action's contribution to the dialog score is not dependent upon its proximity to the end of the task. An action that accepts a proposed item at the beginning of the dialog should be rewarded as much as an action that accepts a proposed item later in the same dialog.

In order for the learning agents to obtain a large enough variety of experiences to fully explore the state space some exploration technique must be used. We chose to use e-greedy action selection in order to achieve this goal. With this approach the dialog agent makes an on policy action choice with probability 1-e and a random valid action choice the rest of the time.

Training both agents simultaneously causes each agent to learn its policy as an optimal response to the opposing agent. This can create problems in the initial stages of training as each agent has an immature policy that is based on little experience. In this situation each of the agents will associate weights with state action pairs based on action choices of the opposing agent that are themselves not well developed. As training progresses the eccentricities of the initial immature policies are perpetuated and the learning process does not converge on an effective dialog policy for either agent.

In order to combat the problem of converging to an effective policy we divided up the agent training process into multiple epochs. Each epoch is composed of a number of training episodes. The initial epsilon value is set to a large value and for each successive epoch the epsilon value for action selection is decreased. With an initially high epsilon value the agents are able to develop a policy that is initially weighted more heavily towards a response to random action selection than the immature policy of the other agent. As the epsilon value decreases, each agent slowly adjusts its learning to be weighted more heavily towards a response to the other agent's policy. This approach allows the agents to develop a minimally coherent dialog policy before beginning to rely too heavily upon the response of the opposing agent.

Utilizing this strategy of continuously decreasing epsilon values we were able to get both agents to converge to an effective and coherent dialog policy. The initial epsilon value was set to 80

4.3 Objective Function

In the reinforcement learning process the objective function provides the dialog agents with feed-back on the success of each dialog. The specification of this function requires input from a human. For our learning specification we crafted a simple function that attempted to model a human perception of a dialog's quality. Our objective function is linear combination of the solution quality (S) and the dialog length (L), taking the form:

$$o(S, I) = w_1 S - w_2 L$$

where w_1 and w_2 are positive constants. As higher values for S and lower values for L indicate better dialogs, we subtract $w_2 L$ from $w_1 S$. Instead of attempting to hand pick the constants in the objective function, we explored the effects of different values, which we report in Section 5.2.

For our experiment we trained the dialog agents for 200 epochs, where each epoch consisted of 200 training episodes. After the training the agents, we then had them perform 5000 dialogs with 100% on-policy action selection (i.e. strictly following the learned policy). The results of these 5000 dialogs were then combined to obtain an average plan score and average number of interactions for the policy of the agents. These two values are then combined according to the objective function to obtain a numeric score for the learned policy.

5 Results

In this section, we present the results of the dialog policies that we learned. We first present 3 baseline policies to which we will compare the performance of our learned policies. We will then present results varying the weights in the objective function in comparison to the baseline policies. As we are learning a pair of policies—one for the system and one representing the user—we explore how well the system policy does against handcrafted ones, that will represent what a user might do, rather than test it against its learned counter-part.

5.1 Baseline Policies

In order to provide comparative data to evaluate the effectiveness of our approach, we will compare the performance of the policies learned for the system and user against several pairs of handcrafted policies. The first pair implement the **unrestricted** initiative strategy of Yang and Heeman. Here, one conversant, A, proposes an item and then the other, B, informs A of any violated preferences. B then proposes an alternative and A informs B of any violated preferences. The process repeats until an item is proposed that does not violate any of the other agent's preferences. The second pair of policies implement the **restricted** initiative policy of Yang and Heeman, in which A proposes an item and B informs A of any violated preferences. However, the conversants do not switch roles: it is always A who proposes items and B that informs of preferences and accepts. These two policies represent successful handcrafted pairs of dialog policies. The third pair represents a minimum performance: A proposes an item and B simply accepts it. This is repeated for all 5 items, with A making all of the proposals. This policy is an **un-collaborative** approach, which represents how well A can do on its own.

5.2 Impact of Weights on Learned Policy

We first explore the ability of the reinforcement learning algorithm to learn a dialog policy pair that is optimal with respect to the objective function. The only important aspect of the weights is the ratio between the two: w_2/w_1. We varied the ratio from 0.1 to 0.5 in increments of 0.02. For each weight setting, we learned 66 policy pairs, and tested each policy pair on 1000 different task configurations. We compared the average objective function score of the learned policy pairs with the baseline restricted policy pair (cf. Scheffler and Young, 2002). Figure 1 shows the percentage of the learned policies that perform at least as well as the unrestricted policy pair

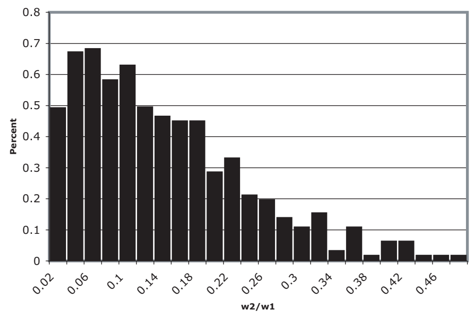


Figure 1: Percentage of learned policies performing better than unrestricted baseline pair.

at each weight setting. Interestingly, it is clear that there is a lack of convergence in the learning process, no weight ratio learns a good policy 100% of the time. Additionally, we see that as the weight ratio increases (putting more emphasis on shorter dialogs), the ability of the algorithm to learn good policies decreases. As the objective function gives this aspect more weight, it is more difficult for the objective function to learn the importance of solution quality. We think this lack of convergence is due to learning both the system and a simulated user at the same time, which is a more difficult reinforcement learning problem than just learning the policy for the system against a fixed user.

5.3 Lack of Convergence

To better understand the lack of convergence, we explore when a single weight is chosen for the objective function. For this analysis, we restricted ourselves to the objective function having a ratio for w_2/w_1 of 0.1, one of the best performing weights from section 5.2. For this setting, we learned a number of policy pairs, each learned from a different sequence of task configurations. We then tested each policy pair on 1000 task configurations, in which actions are selected strictly according to the learned policy. This gives us 1000 dialogs for each policy pair. We then computed the average objective function score for each policy pair and plotted them as a histogram in Figure 2. As can be seen, at this weight setting, 63% of the learned policies achieved an objective function score around 44.8. However, the rest achieved a performance substantially less than this. Hence, the reinforcement learning procedure does not always converge on an optimal solution.

To better understand why reinforcement learning is not always converging, we examined the components of the objective function score: solution quality and dialog length. Figure 3 uses the same x-axis as Figure 2: average objective function score. The y-axis plots the average solution quality and average dialog length. We see that at this weight ratio, all learned dialogue pairs are very consistent in solution quality, but that the difference in objective function scores is mainly due to differences in dialog length. This is consistent with our earlier observation that the reinforcement learning strategy sometimes disproportionately favors shorter dialog length.

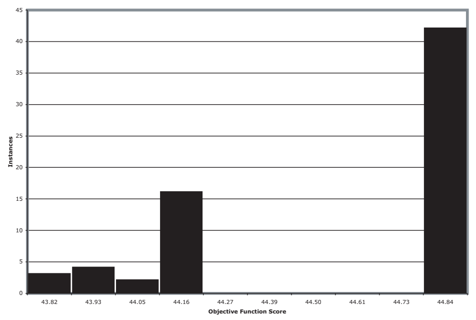


Figure 2: Average objective function scores for policies learned with w_2/w_1 = 0.1.

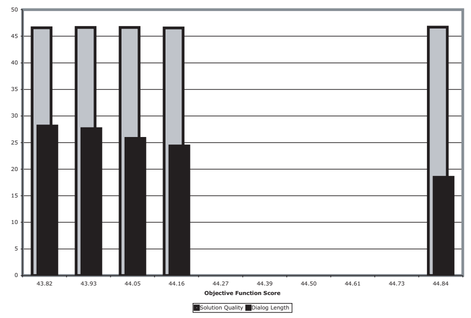


Figure 3: Variation of solution quality and dialogue length versus objective function score for policies learned with w_2/w_1 = 0.1.

5.4 Consistency of Policies

For the weight ratio of 0.1, the reinforcement learning algorithm usually finds a good policy pair. To further improve the likelihood of this happening, we could learn multiple policy pairs, and then pick the best performing one. In this section, we compare learned policies chosen in this way against the restricted baseline pairs. We learned 10 sets of 10 dialogue pairs. We then ran each on 1000 task configurations and chose the best performing policy pair in each set. We then ran the resulting 10 policy pairs on another set of 1000 task configurations. Table 1 gives the average objective function score for each of the 10 learned policy pairs and the 3 baseline pairs. From the table, we see that the learned policy pair performs almost as well as the restricted policy pair, for both solution quality and dialog length.

5.5 Robustness of Learned Policies

All of the results so far have used the learned policy for the system interacting with the corresponding policy that was learned for the user. However, there

	Objective Function	Solution Quality	Dialog Length
Learned Policies	44.90	46.71	18.17
Restricted	45.04	46.89	18.44
Unrestricted	44.40	46.80	24.07
Uncollaborative	32.52	33.62	11.00

Table 1: Comparison of Learned Policies

is no guarantee that a real user will behave like the learned policy. Thus, the true test of our approach is to run the learned system policy against actual users. The problem with testing our policies against actual users is that there are a number of aspects of dialog that we have not modeled, such as non-understandings, misunderstandings, and even parsing sentences into the action specification and generating sentences from the action specification. Thus, as a simplification we tested our learned system policy on the handcrafted baseline policies.

For the weight ratio of 0.1, we learned 10 sets of 10 pairs of policies and choose the best policy pair from each set. For each of the 10 policy pairs, we ran the system policy against the 6 individual policies from the 3 baseline policy pairs. We changed the hand-crafted policies slightly from Yang and Heeman so that the policies would not fail if they encountered unexpected input. For example, for the restricted policy for A (the conversant who proposes but never informs), if the learned policy proposes an item, A always rejects it. For the restricted policy for B (the conversant who informs but never proposes), if the learned policy releases the turn when there is not an item proposed, B simply releases the turn back to the learned policy.

Figure 4 shows the resulting average objective function scores on 1000 dialog runs. For each baseline policy, we show the performance with the policy pair, and then with each side of the baseline policy interacting with the learned policy. We see that although the performance of the learned policy is not as good as with the handcrafted pair, the performance is close, with the major shortcoming being a general increase in dialog length. Thus, the policies that we have learned our robust against different strategies a user might want to use.

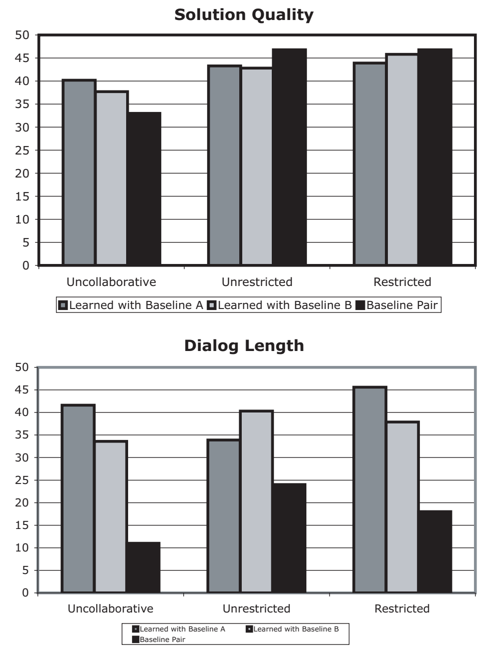


Figure 4: Learned dialogue policies interacting with baseline policies.

6 Conclusion

In this paper, we proposed using reinforcement for learning a dialog strategy for the system. Our approach differs from past research in that we learn the system policy in conjunction with learning a user policy. This approach of learning the user policy allows us to minimize human involvement, as neither a training corpus must be collected nor a simulated user built. Thus, the only human input required for this approach was to define the domain task and to define success in that domain. While our training approach did not always find an effective policy, we overcame this obstacle by carefully choosing a ratio for the weights in the objective function and by running the learning algorithm multiple times. Our approach resulted in learned system and user dialog policies that achieved comparable performance with handcrafted system and user policy pairs. Furthermore, the learned system policies were robust. When the learned system policies ‘conversed’ with the handcrafted user policies, the resulting dialogs had comparable solution quality to what the handcrafted system and user policies achieved together.

Even with the lack of convergence our approach could be applied to more complicated domains in order to learn an effective dialog policy. Our approach would be especially useful in situations where there are no existing corpora of human-human interactions for the domain or as a way to provide a check against a policy based on human intuition. In most situations where the domain requires significant collaboration between the dialog system and the user, training both the system and a user simultaneously will prove to be much less costly and labor intensive approach.

7 Acknowledgments

The authors thank John Moody and Fan Yang for helpful discussions. Partial funding for this research was provided by the National Science Foundation under grant IIS-0326496. The first author is now at Google.

References

K. Georgila, J. Henderson, and O. Lemon. 2005. Learning user simulations for information state update dialogue systems. In *Eurospeech*, Lisbon Portugal.

M. Kearns and S. Singh. 1998. Finite-sample convergence rates for q-learning and indirect algorithms. In *NIPS*, Denver CO.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

K. Scheffler and S. J. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *HLT*, pages 12–18.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing dialogue managment with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press, Cambridge MA.

D. Traum and E. Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599.

M. Walker. 1995. Testing collaborative strategies by computational simulation: Cognitive and task effects. *Knowledge-Based Systems*, 8:105–116.

M. Walker. 2000. An application of reinforcement learning to dialog strategy selection in a spoken dialogue system. *Journal of Artificial Intelligence Research*.

F. Yang and P. Heeman. 2004. Using computer simulation to compare two models of mixed-initiative. In *ICSLP*.

Author Index