NAACL HLT 2007

Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics

Proceedings of the Main Conference

Candace Sidner, General Chair
Tanja Schultz, Matthew Stone, and ChengXiang Zhai
Program Committee Chairs

22–27 April 2007
Rochester, New York, USA

Production and Manufacturing by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53704

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-507-476-0860
acl@aclweb.org

We Thank Our Sponsors

Kodak

Microsoft Research

Powerset
NATURAL LANGUAGE SEARCH

THOMSON ™

AMTA
The Association for Machine Translation in the Americas

IBM®

LANGUAGE WEAVER

Preface from the General Chair

This year the annual conference organized by the North American chapter of the Association for Computational Linguistics (NAACL) has undergone a name change to NAACL HLT. This change reflects the integral part that all of Human Language Technology plays in the NAACL. It is symbolic of the focus of the conference, which is represented by the collection of submitted and accepted papers. They span our community's emphasis on speech processing, information retrieval and language processing techniques and applications.

The yearly NAACL conference is always the result of the volunteer contributions of a great many people from the NAACL community who put in many hours to make the conference possible. Most of the subcommittees of the organizing committee include researchers from the areas of language processing, speech processing and information retrieval, again reflecting the diversity of expertise and interests in the NAACL world.

Each year the general chair calls on a new group of members to serve as the organizing committee. They learn, from scratch, with advice from the previous organizing committee, the tasks needed to make the conference happen. I want to thank each of them for their hard work and good-natured spirit through this process. I thank the program chairs, Tanja Schultz, Matthew Stone, and ChengXiang Zhai; the local arrangement chairs, James Allen, Dan Gildea, and Lenhart Schubert; the tutorial and workshop chairs, James Allan, Marti Hearst, and Gina Levow; the publications chairs, Yang Liu, Ronnie Smith, and Ellen Voorhees; the sponsorship chair David Day, and exhibits chair, Tim Paek; the publicity chairs, Dilek Hakkani-Tür, Miles Osbourne, and Tomek Strzalkowski; the demos chairs, Bob Carpenter, Amanda Stent, and Jason Williams; and the doctoral consortium chairs, Jackson Liscombe, Phil Michalak, and the consortium faculty advisor, Julia Hirschberg.

In additional to the organizing committee, thanks are due to the senior program committee, all the paper reviewers, and the students who volunteered during the conference. A special thank you to our conference sponsors, whose contributions made this conference possible: the Eastman Kodak company, Microsoft Research, Powerset, Thomson, the Association For Machine Translation in the Americas, IBM, and Language Weaver.

Finally, I would also like to thank the NAACL executive committee and the Advisory Board for their advice in preparation for the conference. I especially want to thank Priscilla Rasmussen, who served as Treasurer for this year's meeting, as well as in her normal role as Business Manager for the ACL office. Her knowledge, willingness to make all the ahead-details for the conference go smoothly, and her skills as NAACL corporate memory were invaluable to this conference and its organizers.

Candace L. Sidner
BAE Systems AIT
General Chair, NAACL HLT 2007

Preface from the Program Co-Chairs

We welcome you to NAACL HLT 2007, Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics. NAACL HLT 2007 continues to attract high quality submissions across three broad topic areas – natural language processing, information retrieval, and speech recognition. This year, 298 full papers were submitted and 72 accepted (24% acceptance rate); 150 late-breaking (short) papers were submitted and 55 accepted (37% acceptance rate). The numbers of submissions for both full and short papers continue to grow compared to those of last year.

Reviewing of the submissions was double blind and was handled using a two tiered reviewing system. The PC Chairs selected 30 internationally recognized experts as senior program committee (SPC) members. Each SPC member then selected a group of experts in specific areas to review both the full and short submitted papers. The complete PC numbered around 340. Three (or two in the case of short papers) reviewers and one SPC member were assigned per paper. The SPC oversaw the reviewing process, helped resolve any disputes, and at the end produced, for each paper, an overview of the reviewers' comments along with a preliminary acceptance decision. The final decisions were made by the program chairs based on online discussions among the SPC members.

Three award papers were chosen by the program chairs based on reviews, recommendations, the papers themselves, and our sense that the research efforts epitomize the interactions and opportunities across HLT that the conference aims to foster. The award for the best paper goes to: "Combining Outputs from Multiple Machine Translation Systems" by Antti-Veikko Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan and Bonnie Dorr. The award for the best student paper goes to: "Global, Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming", by Pascal Denis and Jason Baldridge. The award for the best late-breaking news paper goes to "Exploring Affect-Context Dependencies for Adaptive System Development" by Kate Forbes-Riley, Mihai Rotaru, Diane Litman and Joel Tetreault. Congratulations to all the authors!

Reflecting its multi-disciplinary nature, the NAACL HLT 2007 Program consists of oral/poster presentations of full and short papers and software demonstrations that cover a broad spectrum of topics in natural language processing, information retrieval, and speech recognition. We are honored to have two prominent keynote speakers, Franz Josef Och (Google Inc.) and Luis von Ahn (Carnegie Mellon University) for what will undoubtedly be thought-provoking and enjoyable keynote talks. In addition, the program also features a special panel on high impact future research directions for HLT (thanks to Donna Harman).

We are indebted to all the authors who submitted papers to the conference and all those who helped us put together the conference program, especially all the reviewers and SPC members who volunteered their time and worked many long hours reviewing and, later, discussing the submissions. We are also grateful to our General Conference Chair Candy Sidner and Chair of the NAACL Board Owen Rambow for their advice and support, and to Rich Gerber for his help with using the START reviewing system. The NAACL HLT conference has a PC chair for each of its three disciplines. Although work tasks were shared between the three chairs equally, as natural language processing received by far the greatest number of submissions, Matthew Stone ended up having to oversee many more papers and recruit many more SPC members than the other two chairs. He has also taken the primary responsibility of managing the review process and coordinating our effort. Therefore, the two other chairs of NAACL HLT 2007 (Tanja Schultz & ChengXiang Zhai), wish to thank Matthew for all of his additional work in pulling this conference together.

Once again, we welcome you to NAACL HLT 2007 and hope that you enjoy the conference!

Tanja Schultz, Carnegie Mellon University
Matthew Stone, Rutgers University
ChengXiang Zhai, University of Illinois at Urbana-Champaign

Conference Organizers

General Chair:
Candace Sidner, BAE Systems Inc-AIT

Local Arrangements:
James Allen, University of Rochester
Daniel Gildea, University of Rochester
Lenhart Schubert, University of Rochester

Program Committee Chairs:
Tanja Schultz, Carnegie Mellon University
Matthew Stone, Rutgers University
ChengXiang Zhai, University of Illinois at Urbana-Champaign

Publicity Chairs:
Dilek Hakkani-Tür, ICSI
Miles Osbourne, Edinburgh University
Tomek Strzalkowski, SUNY Albany

Workshops and Tutorials Chairs:
James Allan, University of Massachusetts
Marti Hearst, UC Berkeley
Gina Levow, University of Chicago

Demonstrations Chairs:
Bob Carpenter, Alias I
Amanda Stent, SUNY Stony Brook
Jason Williams, AT&T

Exhibits Chair:
Tim Paek, Microsoft

Sponsorship Chair:
David Day, MITRE

Publications:
Yang Liu, UT Dallas
Ronnie Smith, East Carolina University
Ellen Voorhees, NIST

Program Committee Members:

Jeff Adams	Eugene Agichtein	Gregory Aist
Yaser Al-Onaizan	Rie Ando	Galen Andrew
Doug Arnold	Necip Fazil Ayan	Saliha Azzam
Michiel Bacchiani	Jason Baldridge	Timothy Baldwin
Srinivas Bangalore	Michele Banko	Regina Barzilay
Frederic Bechet	Linda Bell	Robert Belvin
Paul Bennett	Shane Bergsma	Nicola Bertoldi
Daniel Bikel	John Blitzer	Dan Bohus
Gary Borchardt	Johan Bos	Chris Brew
Eric Brown	Ralf Brown	Sabine Buchholz
Miriam Butt	Aoife Cahill	Ellen Campana
Yunbo Cao	Jaime Carbonell	Violetta Cavalli-Sforza
Joyce Chai	Jason Chang	Eugene Charniak
Ciprian Chelba	Hsin-Hsi Chen	Jiangping Chen
Colin Cherry	David Chiang	Timothy Chklovski
Ken Church	Alexander Clark	Robert Clark
Stephen Clark	William Cohen	Trevor Cohn
Michael Collins	Alistair Conkie	Paul Cook
Ann Copestake	Mark Core	Richard Crouch
Berthold Crysmann	Silviu-Petru Cucerzan	Aron Culotta
James Curran	Robert Dale	Hoa Dang
Hal Daume	Renato De Mori	Nikhil Dinesh
Christy Doran	John Dowding	Amit Dubey
Kevin Duh	Jakob Elming	Katrin Erk
David Farwell	Afsaneh Fazly	Junlan Feng
Dan Flickinger	Martin Forst	Mary Ellen Foster
George Foster	Alexander Fraser	Robert Frederking
Andrew Freeman	Dayne Freitag	Mark Fuhs
Pascale Fung	Sadaoki Furui	Michel Galley
Kallirroi Georgila	Roxana Girju	Gregory Grefenstette
Michelle Gregory	Teg Grenager	Stephanie Haas
Dilek Hakkani-Tur	Keith Hall	Jeff Hancock
Donna Harman	Mary Harper	Anthony Hartley
Xiaodong He	James Henderson	Andrew Hickl
Graeme Hirst	Liang Huang	Chu-Ren Huang
Diana Inkpen	Kentaro Inui	Pierre Isabelle
Martin Jansche	Mark Johnson	Kristiina Jokinen
Gareth Jones	Rosie Jones	Joemon Jose
Aravind Joshi	Stan Jou	Jeremy Kahn
Nanda Kambhatla	Min Yen Kan	Lauri Karttunen
Boris Katz	John Kelleher	Frank Keller
Sanjeev Khudanpur	Simon King	Tracy King

Program Committee Members (continued):

Katrin Kirchoff	Esther Klabbers	Alexandre Klementiev
Kevin Knight	Greg Kondrak	Valia Kordoni
Anna Korhonen	Sandra Kuebler	Jonas Kuhn
Shankar Kumar	Mikko Kurimo	KL Kwok
Namhee Kwon	Mounia Lalmas	Ian Lane
Philippe Langlaise	Guy Lapalme	Mirella Lapata
Ray Larson	Staffan Larsson	Alex Lascarides
Alon Lavie	Matthew Lease	Gary Lee
Jong-Hyeok Lee	Lillian Lee	Oliver Lemon
Gina Levow	Roger Levy	Hang Li
Xiaolong Li	Chin-Yew Lin	Jimmy Lin
Yang Liu	Deryle Lonsdale	Adam Lopez
Jie Lu	Xiaoqiang Luo	Dick Lyon
Bill MacCartney	Elliott Macklovitch	Bente Maegaard
Bernardo Magnini	Rob Malouf	Gideon Mann
Chris Manning	Daniel Marcu	Evgeny Matusov
Nicolas Maudet	Andrew McCallum	Diana McCarthy
Kathleen McKeown	Paul McNamee	Dan Melamed
Arul Menezes	Donald Metzler	Rada Mihalcea
Teruko Mitamura	Yusuke Miyao	Mehryar Mohri
Dan Moldovan	Christof Monz	Alessandro Moschitti
Srini Narayanan	Mark-Jan Nederhof	Hwee Tou Ng
Vincent Ng	Sergei Nirenburg	Joakim Nivre
Stephan Oepen	Iadh Ounis	Martha Palmer
Bo Pang	Rebecca Passoneau	Fuchun Peng
Gerald Penn	Marco Pennacchiotti	Paul Piwek
Sameer Pradhan	John Prange	Partha Pratim Talukdar
Stephen Pulman	Chris Quirk	Dragomir Radev
Owen Rambow	Deepak Ravichandran	Florence Reeder
David Reitter	Giuseppe Riccardi	Stephen Richardson
Korin Richmond	Sebastian Riedel	Ellen Riloff
Jin Rong	Salim Roukos	Fatiha Sadat
Kenji Sagae	Mark Sanderson	Murat Saraclar
Ruhi Sarikaya	Thomas Schaaf	Charles Schafer
Helmut Schmid	Hinrich Schuetze	Sabine Schulte im Walde
Satoshi Sekine	Zak Shafran	James Shaw
Libin Shen	Chilin Shih	Luo Si
Malcolm Slaney	Noah Smith	Ronnie Smith
David Smith	Rion Snow	Stephen Soderland
Thamar Solorio	Harold Somers	Dawei Song
Radu Soricut	Richard Sproat	Mark Steedman
Amanda Stent	Laura Stoia	Nicola Stokes
Michael Strube	Tomek Strzalkowski	Jian Su
Mihai Surdeanu	Charles Sutton	Ann Syrdal

Program Committee Members (continued):

Joel Tetreault	Simone Teufel	Stefan Thater
Ye Tian	Christoph Tillmann	Tomoki Toda
Takenobu Tokunaga	Elaine Toms	Arthur Toth
Kristina Toutanova	Peter Turney	Andrew Turpin
Nicola Ueffing	Kees Van Deemter	Lucy Vanderwende
Sebastian Varges	Renata Vieira	David Vilar
Stephan Vogel	Ellen Voorhees	Ye-Yi Wang
Karen Ward	Taro Watanabe	Andy Way
Bonnie Webber	Eric Wehrli	Ben Wellington
Fuliang Weng	Martin Westphal	Ed Whittaker
Janyce Wiebe	Jason Williams	Shuly Wintner
Dekai Wu	Fei Xia	David Yarowsky
Alex Yates	Chen Yu	Hugo Zaragoza
Daniel Zeman	Richard Zens	Jian Zhang
Joy Zhang	Yi Zhang	Bing Zhao
Liang Zhou	Ming Zhou	Imed Zitouni

Table of Contents

Conference Program

11:10-12:25	Paper sessions
	Text Classification
11:10-11:35	*Automating Creation of Hierarchical Faceted Metadata Structures* Emilia Stoica, Marti Hearst and Megan Richardson
11:35-12:00	*Cross-Instance Tuning of Unsupervised Document Clustering Algorithms* Damianos Karakos, Jason Eisner, Sanjeev Khudanpur and Carey Priebe
12:00-12:25	*Using "Annotator Rationales" to Improve Machine Learning for Text Categorization* Omar Zaidan, Jason Eisner and Christine Piatko
	Conversational Systems
11:10-11:35	*Combining Reinformation Learning with Information-State Update Rules* Peter Heeman
11:35-12:00	*Estimating the Reliability of MDP Policies: a Confidence Interval Approach* Joel Tetreault, Dan Bohus and Diane Litman
12:00-12:25	*An Exploration of Eye Gaze in Spoken Language Processing for Multimodal Conversational Interfaces* Shaolin Qu and Joyce Chai
	Extracting Sentiment
11:10-11:35	*Extracting Semantic Orientations of Phrases from Dictionary* Hiroya Takamura, Takashi Inui and Manabu Okumura
11:35-12:00	*Multiple Aspect Ranking Using the Good Grief Algorithm* Benjamin Snyder and Regina Barzilay
12:00-12:25	*Extracting Appraisal Expressions* Kenneth Bloom, Navendu Garg and Shlomo Argamon
12:25-2:00	Lunch

Tuesday, April 24, 2007 (continued)

3:00-3:15 *Simultaneous Identification of Biomedical Named-Entity and Functional Relation Using Statistical Parsing Techniques*
Zhongmin Shi, Anoop Sarkar and Fred Popowich

3:15-3:30 *Chinese Named Entity Recognition with Cascaded Hybrid Model*
Xiaofeng Yu

3:30-3:45 *Entity Extraction is a Boring Solved Problem—Or is it?*
Marc Vilain, Jennifer Su and Suzi Lubar

3:45-4:00 *A Semi-Automatic Evaluation Scheme: Automated Nuggetization for Manual Annotation*
Liang Zhou, Namhee Kwon and Eduard Hovy

Speech and IR

2:00-2:15 *Joint versus Independent Phonological Feature Models Within CRF Phone Recognition*
Ilana Bromberg, Jeremy Morris and Eric Fosler-Lussier

2:15-2:30 *On Using Articulatory Features for Discriminative Speaker Adaptation*
Florian Metze

2:30-2:45 *Reversible Sound-to-Letter/Letter-to-Sound Modeling Based on Syllable Structure*
Stephanie Seneff

2:45-3:00 *iROVER: Improving System Combination with Classification*
Dustin Hillard, Bjoern Hoffmeister, Mari Ostendorf, Ralf Schlueter and Hermann Ney

3:00-3:15 *Advances in the CMU/Interact Arabic GALE Transcription System*
Mohamed Noamany, Thomas Schaaf and Tanja Schultz

3:15-3:30 *Are Some Speech Recognition Errors Easier to Detect than Others?*
Yongmei Shi and Lina Zhou

3:30-3:45 *Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System*
Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye and Steve Young

3:45-4:00 *Document Similarity Measures to Distinguish Native vs. Non-Native Essay Writers*
Olga Gurevich and Paul Deane

4:00-4:30 Break

Wednesday, April 25, 2007

9:00-10:00	Invited talk by Luis von Ahn, Carnegie Mellon University
10:00-10:30	Break
10:30-11:15	Panel: MINDS: High Impact Future Research Directions for HLT
11:15-12:00	NAACL Business Meeting
12:00-1:45	Lunch
1:45-3:25	Paper sessions
	Parsing
1:45-2:10	*Tree Revision Learning for Dependency Parsing* Giuseppe Attardi and Massimiliano Ciaramita
2:10-2:35	*Incremental Non-Projective Dependency Parsing* Joakim Nivre
2:35-3:00	*Improved Inference for Unlexicalized Parsing* Slav Petrov and Dan Klein
3:00-3:25	*Approximate Factoring for A* Search* Aria Haghighi, John DeNero and Dan Klein
	Semantics and Discourse
1:45-2:10	*A Cascaded Machine Learning Approach to Interpreting Temporal Expressions* David Ahn, Joris van Rantwijk and Maarten de Rijke
2:10-2:35	*Building and Refining Rhetorical-Semantic Relation Models* Sasha Blair-Goldensohn, Kathleen McKeown and Owen Rambow
2:35-3:00	*A Unified Local and Global Model for Discourse Coherence* Micha Elsner, Joseph Austerweil and Eugene Charniak
3:00-3:25	*Randomized Decoding for Selection-and-Ordering Problems* Pawan Deshpande, Regina Barzilay and David Karger

Exploiting acoustic and syntactic features for prosody labeling in a maximum entropy framework

Vivek Rangarajan, Shrikanth Narayanan
Speech Analysis and Interpretation Laboratory
University of Southern California
Viterbi School of Electrical Engineering
vrangara@usc.edu,shri@sipi.usc.edu

Srinivas Bangalore
AT&T Research Labs
180 Park Avenue
Florham Park, NJ 07932, U.S.A.
srini@research.att.com

Abstract

In this paper we describe an automatic prosody labeling framework that exploits both language and speech information. We model the syntactic-prosodic information with a maximum entropy model that achieves an accuracy of 85.2% and 91.5% for pitch accent and boundary tone labeling on the Boston University Radio News corpus. We model the acoustic-prosodic stream with two different models, one a maximum entropy model and the other a traditional HMM. We finally couple the syntactic-prosodic and acoustic-prosodic components to achieve significantly improved pitch accent and boundary tone classification accuracies of 86.0% and 93.1% respectively. Similar experimental results are also reported on Boston Directions corpus.

1 Introduction

Prosody refers to intonation, rhythm and lexical stress patterns of spoken language that convey linguistic and paralinguistic information such as emphasis, intent, attitude and emotion of a speaker. Prosodic information associated with a unit of speech, say, syllable, word, phrase or clause, influence all the segments of the unit in an utterance. In this sense they are also referred to as suprasegmentals (Lehiste, 1970). Prosody in general is highly dependent on individual speaker style, gender, dialect and other phonological factors. The difficulty in reliably characterizing suprasegmental information present in speech has resulted in symbolic and parameteric prosody labeling standards like ToBI (Tones and Break Indices) (Silverman et al., 1992) and Tilt model (Taylor, 1998) respectively.

Prosody in spoken language can be characterized through acoustic features or lexical features or both. Acoustic correlates of duration, intensity and pitch, like syllable nuclei duration, short time energy and fundamental frequency (f0) are some acoustic features that are perceived to confer prosodic prominence or stress in English. Lexical features like parts-of-speech, syllable nuclei identity, syllable stress of neighboring words have also demonstrated high degree of discriminatory evidence in prosody detection tasks.

The interplay between acoustic and lexical features in characterizing prosodic events has been successfully exploited in text-to-speech synthesis (Bulyko and Ostendorf, 2001; Ma et al., 2003), speech recognition (Hasegawa-Johnson et al., 2005) and speech understanding (Wightman and Ostendorf, 1994). Text-to-speech synthesis relies on lexical features derived predominantly from the input text to synthesize natural sounding speech with appropriate prosody. In contrast, output of a typical automatic speech recognition (ASR) system is noisy and hence, the acoustic features are more useful in predicting prosody than the hypothesized lexical transcript which may be erroneous. Speech understanding systems model both the lexical and acoustic features at the output of an ASR to improve natural language understanding. Another source of renewed interest has come from spoken language translation (Nöth et al., 2000; Agüero et al., 2006). A prerequisite for all these applications is accurate prosody detection, the topic of the present work.

In this paper, we describe our framework for building an automatic prosody labeler for English. We report results on the Boston University (BU) Radio Speech Corpus (Ostendorf et al., 1995) and Boston Directions Corpus (BDC) (Hirschberg and Nakatani, 1996), two publicly available speech corpora with manual ToBI annotations intended for experiments in automatic prosody labeling. We condition prosody not only on word strings and their parts-of-speech but also on richer syntactic information encapsulated in the form of Supertags (Bangalore and Joshi, 1999). We propose a maximum entropy modeling framework for the syntactic features. We model the acoustic-prosodic stream with two different models, a maximum entropy model and a more traditional hidden markov model (HMM). In an automatic prosody labeling task, one is essentially try-

Proceedings of NAACL HLT 2007, pages 1–8,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

ing to predict the correct prosody label sequence for a given utterance and a maximum entropy model offers an elegant solution to this learning problem. The framework is also robust in the selection of discriminative features for the classification problem. So, given a word sequence $W = \{w_1, \cdots, w_n\}$ and a set of acoustic-prosodic features $A = \{o_1, \cdots, o_T\}$, the best prosodic label sequence $L^* = \{l_1, l_2, \cdots, l_n\}$ is obtained as follows,

$$\begin{aligned} L^* &= \arg\max_L P(L|A, W) &(1)\\ &= \arg\max_L P(L|W).P(A|L, W) &(2)\\ &\approx \arg\max_L P(L|\Phi(W)).P(A|L, W) &(3)\end{aligned}$$

where $\Phi(W)$ is the syntactic feature encoding of the word sequence W. The first term in Equation (3) corresponds to the probability obtained through our maximum entropy syntactic model. The second term in Equation (3), computed by an HMM corresponds to the probability of the acoustic data stream which is assumed to be dependent only on the prosodic label sequence.

The paper is organized as follows. In section 2 we describe related work in automatic prosody labeling followed by a description of the data used in our experiments in section 3. We present prosody prediction results from off-the-shelf synthesizers in section 4. Section 5 details our proposed maximum entropy syntactic-prosodic model for prosody labeling. In section 6, we describe our acoustic-prosodic model and discuss our results in section 7. We finally conclude in section 8 with directions for future work.

2 Related work

Automatic prosody labeling has been an active research topic for over a decade. Wightman and Ostendorf (Wightman and Ostendorf, 1994) developed a decision-tree algorithm for labeling prosodic patterns. The algorithm detected phrasal prominence and boundary tones at the syllable level. Bulyko and Ostendorf (Bulyko and Ostendorf, 2001) used a prosody prediction module to synthesize natural speech with appropriate prosody. Verbmobil (Nöth et al., 2000) incorporated prosodic labeling into a translation framework for improved linguistic analysis and speech understanding.

Prosody has typically been represented either symbolically, e.g., ToBI (Silverman et al., 1992) or parametrically, e.g., Tilt Intonation Model (Taylor, 1998). Parametric approaches either restrict the variants of prosody by definition or automatically learn prosodic patterns from data (Agüero et al., 2006). The BU corpus is a widely used corpus with symbolic representation of prosody. The hand-labeled ToBI annotations make this an attractive corpus to perform prosody labeling experiments. The main drawback of this corpus is that it comprises only read speech. Prosody labeling on spontaneous speech corpora like Boston Directions corpus (BDC), Switchboard (SWBD) has garnered attention in (Hirschberg and Nakatani, 1996; Gregory and Altun, 2004).

Automatic prosody labeling has been achieved through various machine learning techniques, such as decision trees (Hirschberg, 1993; Wightman and Ostendorf, 1994; Ma et al., 2003), rule-based systems (Shimei and McKeown, 1999), bagging and boosting on CART (Sun, 2002), hidden markov models (Conkie et al., 1999), neural networks (Hasegawa-Johnson et al., 2005),maximum-entropy models (Brenier et al., 2005) and conditional random fields (Gregory and Altun, 2004).

Prosody labeling of the BU corpus has been reported in many studies (Hirschberg, 1993; Hasegawa-Johnson et al., 2005; Ananthakrishnan and Narayanan, 2005). Hirschberg (Hirschberg, 1993) used a decision-tree based system that achieved 82.4% speaker dependent accent labeling accuracy at the word level on the BU corpus using lexical features. (Ross and Ostendorf, 1996) also used an approach similar to (Wightman and Ostendorf, 1994) to predict prosody for a TTS system from lexical features. Pitch accent accuracy at the word-level was reported to be 82.5% and syllable-level accent accuracy was 80.2%. (Hasegawa-Johnson et al., 2005) proposed a neural network based syntactic-prosodic model and a gaussian mixture model based acoustic-prosodic model to predict accent and boundary tones on the BU corpus that achieved 84.2% accuracy in accent prediction and 93.0% accuracy in intonational boundary prediction. With syntactic information alone they achieved 82.7% and 90.1% for accent and boundary prediction, respectively. (Ananthakrishnan and Narayanan, 2005) modeled the acoustic-prosodic information using a coupled hidden markov model that modeled the asynchrony between the acoustic streams. The pitch accent and boundary tone detection accuracy at the syllable level were 75% and 88% respectively. Our proposed maximum entropy syntactic model outperforms previous work. On the BU corpus, with syntactic information alone we achieve pitch accent and boundary tone accuracy of 85.2% and 91.5% on the same training and test sets used in (Chen et al., 2004; Hasegawa-Johnson et al., 2005). Further, the coupled model with both acoustic and syntactic information results in accuracies of 86.0% and 93.1% respectively. On the BDC corpus, we achieve pitch accent and boundary tone accuracies of 79.8% and 90.3%.

3 Data

The BU corpus consists of broadcast news stories including original radio broadcasts and laboratory sim-

	BU				**BDC**			
Corpus statistics	**f2b**	**f1a**	**m1b**	**m2b**	**h1**	**h2**	**h3**	**h4**
# Utterances	165	69	72	51	10	9	9	9
# words (w/o punc)	12608	3681	5058	3608	2234	4127	1456	3008
# pitch accents	6874	2099	2706	2016	1006	1573	678	1333
# boundary tones (w IP)	3916	1059	1282	1023	498	727	361	333
# boundary tones (w/o IP)	2793	684	771	652	308	428	245	216

Table 1: BU and BDC dataset used in experiments

ulations recorded from seven FM radio announcers. The corpus is annotated with orthographic transcription, automatically generated and hand-corrected part-of-speech tags and automatic phone alignments. A subset of the corpus is also hand annotated with ToBI labels. In particular, the experiments in this paper are carried out on 4 speakers similar to (Chen et al., 2004), 2 male and 2 female referred to hereafter as **m1b**, **m2b**, **f1a** and **f2b**. The BDC corpus is made up of elicited monologues produced by subjects who were instructed to perform a series of direction-giving tasks. Both spontaneous and read versions of the speech are available for four speakers **h1**, **h2**, **h3** and **h4** with hand-annotated ToBI labels and automatic phone alignments, similar to the BU corpus. Table 1 shows some of the statistics of the speakers in the BU and BDC corpora.

In Table 1, the pitch accent and boundary tone statistics are obtained by decomposing the ToBI labels into binary classes using the mapping shown in Table 2.

BU Labels	**Intermediate Mapping**	**Coarse Mapping**
H*,!H* L* *,*?,X*?	Single Accent	**accent**
H+!H*,L+H*,L+!H* L*+!H,L*+H	Bitonal Accent	
L-L%,!H-L%,H-L% H-H% L-H% %?,X%?,%H	Final Boundary tone	**btone**
L-,H-,!H- -X?,-?	Intermediate Phrase (IP) boundary	
<,>,no label	**none**	**none**

Table 2: ToBI label mapping used in experiments

In all our prosody labeling experiments we adopt a leave-one-out speaker validation similar to the method in (Hasegawa-Johnson et al., 2005) for the four speakers with data from one speaker for testing and from the other three for training. For the BU corpus, **f2b** speaker was always used in the training set since it contains the most data. In addition to performing experiments on all the utterances in BU corpus, we also perform identical experiments on the train and test sets reported in (Chen et al., 2004) which is referred to as Hasegawa-Johnson et al. set.

4 Baseline Experiments

We present three baseline experiments. One is simply based on chance where the majority class label is predicted. The second is a baseline only for pitch accents derived from the lexical stress obtained through look-up from a pronunciation lexicon labeled with stress. Finally, the third and more concrete baseline is obtained through prosody detection in current speech synthesis systems.

4.1 Prosody labels derived from lexical stress

Pitch accents are usually carried by the stressed syllable in a particular word. Lexicons with phonetic transcription and lexical stress are available in many languages. Hence, one can use these lexical stress markers within the syllables and evaluate the correlation with pitch accents. Eventhough the lexicon has a closed vocabulary, letter-to-sound rules can be derived from it for unseen words. For each word carrying a pitch accent, we find the particular syllable where the pitch accent occurs from the manual annotation. For the same syllable, we predict pitch accent based on the presence or absence of a lexical stress marker in the phonetic transcription. The results are presented in Table 3.

4.2 Prosody labeling with Festival and AT&T Natural Voices® Speech Synthesizer

Festival (Black et al., 1998) and AT&T Natural Voices® (NV) speech synthesizer (att,) are two publicly available speech synthesizers that have a prosody prediction module available. We performed automatic prosody labeling using the two synthesizers to get a baseline.

4.2.1 AT&T Natural Voices® Speech Synthesizer

The AT&T NV® speech synthesizer is a half phone speech synthesizer. The toolkit accepts an input text utterance and predicts appropriate ToBI pitch accent and boundary tones for each of

Corpus	Speaker Set	Prediction Module	Pitch accent Chance	Pitch accent Accuracy	Boundary tone Chance	Boundary tone Accuracy
BU	Entire Set	**Lexical stress**	54.33	72.64	-	-
		AT&T Natural Voices	54.33	81.51	81.14	89.10
		Festival	54.33	69.55	81.14	89.54
	Hasegawa-Johnson et al. set	**Lexical stress**	56.53	74.10	-	-
		AT&T Natural Voices	56.53	81.73	82.88	89.67
		Festival	56.53	68.65	82.88	90.21
BDC	Entire Set	**Lexical stress**	57.60	67.42	-	-
		AT&T Natural Voices	57.60	68.49	88.90	84.90
		Festival	57.60	64.94	88.90	85.17

Table 3: Classification results of pitch accents and boundary tones (in %) using Festival and AT&T NV® synthesizer

the selected units (in this case, a pair of phones) from the database. We reverse mapped the selected half phone units to words, thus obtaining the ToBI labels for each word in the input utterance. The toolkit uses a rule-based procedure to predict the ToBI labels from lexical information. The pitch accent labels predicted by the toolkit are $L_{\mathbf{accent}} \epsilon \{\mathbf{H*}, \mathbf{L*}, \mathbf{none}\}$ and the boundary tones are $L_{\mathbf{btone}} \epsilon \{\mathbf{L\text{-}L\%}, \mathbf{H\text{-}H\%}, \mathbf{L\text{-}H\%}, \mathbf{none}\}$.

4.2.2 Festival Speech Synthesizer

Festival (Black et al., 1998) is an open-source unit selection speech synthesizer. The toolkit includes a CART-based prediction system that can predict ToBI pitch accents and boundary tones for the input text utterance. The pitch accent labels predicted by the toolkit are $L_{\mathbf{accent}} \epsilon \{\mathbf{H*}, \mathbf{L+H*}, \mathbf{!H*}, \mathbf{none}\}$ and the boundary tones are $L_{\mathbf{btone}} \epsilon \{\mathbf{L\text{-}L\%}, \mathbf{H\text{-}H\%}, \mathbf{L\text{-}H\%}, \mathbf{none}\}$. The prosody labeling results obtained through both the speech synthesis engines are presented in Table 3. The chance column in Table 3 is obtained by predicting the most frequent label in the data set.

In the next sections, we describe our proposed maximum entropy based syntactic model and HMM based acoustic-prosodic model for automatic prosody labeling.

5 Syntactic-prosodic Model

We propose a maximum entropy approach to model the words, syntactic information and the prosodic labels as a sequence. We model the prediction problem as a classification task as follows: given a sequence of words w_i in a sentence $W = \{w_1, \cdots, w_n\}$ and a prosodic label vocabulary ($l_i \epsilon \mathcal{L}$), we need to predict the best prosodic label sequence $L^* = \{l_1, l_2, \cdots, l_n\}$. We approximate the conditional probability to be within a bounded n-gram context. Thus,

$$L^* = \arg\max_L P(L|W,T,S) \quad (4)$$

$$\approx \arg\max_L \prod_i^n p(l_i|w_{i-k}^{i+k}, t_{i-k}^{i+k}, s_{i-k}^{i+k}) \quad (5)$$

where $W = \{w_1, \cdots, w_n\}$ is the word sequence and $T = \{t_1, \cdots, t_n\}$, $S = \{s_1, \cdots, s_n\}$ are the corresponding part-of-speech and additional syntactic information sequences. The variable k controls the context.

The BU corpus is automatically labeled (and hand-corrected) with part-of-speech (POS) tags. The POS inventory is the same as the Penn treebank which includes 47 POS tags: 22 open class categories, 14 closed class categories and 11 punctuation labels. We also automatically tagged the utterances using the AT&T POS tagger. The POS tags were mapped to function and content word categories [1] which was added as a discrete feature. In addition to the POS tags, we also annotate the utterance with Supertags (Bangalore and Joshi, 1999). Supertags encapsulate predicate-argument information in a local structure. They are composed with each other using substitution and adjunction operations of Tree-Adjoining Grammars (TAGs) to derive a dependency analysis of an utterance and its predicate-argument structure. Even though there is a potential to exploit the dependency structure between supertags and prosody labels as demonstrated in (Hirschberg and Rambow, 2001), for this paper we use only the supertag labels.

Finally, we generate one feature vector ($\mathbf{\Phi}$) for each word in the data set (with local contextual features). The best prosodic label sequence is then,

$$L^* = \arg\max_L \prod_i^n P(l_i|\mathbf{\Phi}) \quad (6)$$

To estimate the conditional distribution $P(l_i|\mathbf{\Phi})$ we use the general technique of choosing the maximum entropy (maxent) distribution that estimates the average of each feature over the training data (Berger et al., 1996). This can be written in terms of Gibbs distribution parameterized with weights λ, where V is the size of the prosodic label set. Hence,

$$P(l_i|\mathbf{\Phi}) = \frac{e^{\lambda_{l_i} \cdot \mathbf{\Phi}}}{\sum_{l=1}^{V} e^{\lambda_{l_i} \cdot \mathbf{\Phi}}} \quad (7)$$

[1]function and content word features were obtained through a look-up table based on POS

Corpus	Speaker Set	Syntactic features	k=3	
			accent	**btone**
BU	Entire Set	correct POS tags	84.75	91.39
		AT&T POS + supertags	84.59	91.34
		Joint Model (w AT&T POS + supertags)	84.60	91.36
	Hasegawa-Johnson et al. set	correct POS tags	85.22	91.33
		AT&T POS + supertags	84.95	91.21
		Joint Model (w AT&T POS + supertags)	84.78	91.54
BDC	Entire Set	AT&T POS + supertags	79.81	90.28
		Joint Model (w AT&T POS + supertags)	79.57	89.76

Table 4: Classification results (%) of pitch accents and boundary tones for different syntactic representation ($k = 3$)

We use the machine learning toolkit LLAMA (Haffner, 2006) to estimate the conditional distribution using maxent. LLAMA encodes multiclass maxent as binary maxent to increase the training speed and to scale the method to large data sets. Each of the V classes in the label set $\mathcal{L}$ is encoded as a bit vector such that, in the vector for class i, the i^{th} bit is one and all other bits are zero. Finally, V one-versus-other binary classifiers are used as follows.

$$P(y|\mathbf{\Phi}) = 1 - P(\bar{y}|\mathbf{\Phi}) = \frac{e^{\lambda_y \cdot \mathbf{\Phi}}}{e^{\lambda_y \cdot \mathbf{\Phi}} + e^{\lambda_{\bar{y}} \cdot \mathbf{\Phi}}} \quad (8)$$

where $\lambda_{\bar{y}}$ is the parameter vector for the anti-label $\bar{y}$. To compute $P(l_i|\mathbf{\Phi})$, we use the class independence assumption and require that $y_i = 1$ and for all $j \neq i, y_j = 0$.

$$P(l_i|\mathbf{\Phi}) = P(y_i|\mathbf{\Phi}) \prod_{j \neq i}^{V} P(y_j|\mathbf{\Phi}) \quad (9)$$

5.1 Joint Modeling of Accents and Boundary Tones

Prosodic prominence and phrasing can also be viewed as joint events occurring simultaneously. Previous work by (Wightman and Ostendorf, 1994) suggests that a joint labeling approach may be more beneficial in prosody labeling. In this scenario, we treat each word to have one of the four labels $l_i \ \epsilon \ \mathcal{L}$ = {**accent-btone, accent-none, none-btone, none-none**}. We trained the classifier on the joint labels and then computed the error rates for individual classes. The results of prosody prediction using the set of syntactic-prosodic features for $k = 3$ is shown in Table 4. The joint modeling approach provides a marginal improvement in the boundary tone prediction but is slightly worse for pitch accent prediction.

5.2 Supertagger performance on Intermediate Phrase boundaries

Perceptual experiments have indicated that inter-annotator agreement for ToBI intermediate phrase boundaries is very low compared to full-intonational boundaries (Syrdal and McGory, 2000). Intermediate phrasing is important in TTS applications to synthesize appropriate short pauses to make the utterance sound natural. The significance of syntactic features in the boundary tone prediction prompted us to examine the effect of predicting intermediate phrase boundaries in isolation. It is intuitive to expect supertags to perform well in this task as they essentially form a local dependency analysis on an utterance and provide an encoding of the syntactic phrasal information. We performed this task as a three way classification where $l_i \ \epsilon \ \mathcal{L}$ = {**btone, ip, none**}. The results of the classifier on IPs is shown in Table 5.

Model	Syntactic features	IP accuracy
k=2 (bigram context)	correct POS tags	83.25
	AT&T POS tags	83.32
	supertags	83.37
k=3 (trigram context)	correct POS tags	83.30
	AT&T POS tags	83.46
	supertags	83.74

Table 5: Accuracy (in %) obtained by leave-one out speaker validation using IPs as a separate class on entire speaker set

6 Acoustic-prosodic model

We propose two approaches to modeling the acoustic-prosodic features for prosody prediction. First, we propose a maximum entropy framework similar to the syntactic model where we quantize the acoustic features and model them as discrete sequences. Second, we use a more traditional approach where we train continuous observation density HMMs to represent pitch accents and boundary tones. We first describe the features used in the acoustic modeling followed by a more detailed description of the acoustic-prosodic model.

6.1 Acoustic-prosodic features

The BU corpus contains the corresponding acoustic-prosodic feature file for each utterance. The f0, RMS energy (e) of the utterance along with features for

			Pitch accent		Boundary tone	
Corpus	Speaker Set	Model	Acoustics	Acoustics+syntax	Acoustics	Acoustics+syntax
BU	Entire Set	Maxent acoustic model	80.09	84.53	84.10	91.56
		HMM acoustic model	70.58	85.13	71.28	92.91
	Hasegawa-Johnson et al. set	Maxent acoustic model	80.12	84.84	82.70	91.76
		HMM acoustic model	71.42	86.01	73.43	93.09
BDC	Entire Set	Maxent acoustic model	74.51	78.64	83.53	90.49

Table 6: Classification results of pitch accents and boundary tones (in %) with acoustics only and acoustics+syntax using both our models

distinction between voiced/unvoiced segment, cross-correlation values at estimated f0 value and ratio of first two cross correlation values are computed over 10 msec frame intervals. In our experiments, we use these values rather than computing them explicitly which is straightforward with most audio toolkits. Both the energy and the f0 levels were normalized with speaker specific means and variances. Delta and acceleration coefficients were also computed for each frame. The final feature vector is 6-dimensional comprising of f0, Δf0, Δ^2f0, e, Δe, Δ^2e per frame.

6.2 Maximum Entropy acoustic-prosodic model

We propose a maximum entropy modeling framework to model the continuous acoustic-prosodic observation sequence as a discrete sequence through the means of quantization. The quantized acoustic stream is then used as a feature vector and the conditional probabilities are approximated by an n-gram model. This is equivalent to reducing the vocabulary of the acoustic-prosodic features and hence offers better estimates of the conditional probabilities. Such an n-gram model of quantized continuous features is similar to representing the set of features with a linear fit as done in the tilt intonational model (Taylor, 1998).

The quantized acoustic-prosodic feature stream is modeled with a maxent acoustic-prosodic model similar to the one described in section 5. Finally, we append the syntactic and acoustic features to model the combined stream with the maxent acoustic-syntactic model, where the objective criterion for maximization is Equation (1). The pitch accent and boundary tone prediction accuracies for quantization performed by considering only the first decimal place is reported in Table 6. As expected, we found the classification accuracy to drop with increasing number of bins used in the quantization due to the small amount of training data.

6.3 HMM acoustic-prosodic model

We also investigated the traditional HMM approach to model the high variability exhibited by the acoustic-prosodic features. First, we trained separate context independent single state Gaussian mixture density HMMs for pitch accents and boundary tones in a generative framework. The label sequence was decoded using the viterbi algorithm. Next, we trained HMMs with 3 state left-to-right topology with uniform segmentation. The segmentations need to be uniform due to lack of an acoustic-prosodic model trained on the features pertinent to our task to obtain forced segmentation.

The final label sequence using the maximum entropy syntactic-prosodic model and the HMM based acoustic-prosodic model was obtained by combining the syntactic and acoustic probabilities shown in Equation (3). The syntactic-prosodic maxent model outputs a posterior probability for each class per word. We formed a lattice out of this structure and composed it with the lattice generated by the HMM acoustic-prosodic model. The best path was chosen from the composed lattice through a Viterbi search. The acoustic-prosodic probability $P(A|L,W)$ was raised by a power of γ to adjust the weighting between the acoustic and syntactic model. The value of γ was chosen as 0.008 and 0.015 for pitch accent and boundary tone respectively, by tuning on the training set. The results of the acoustic-prosodic model and the coupled model are shown in Table 6.

7 Discussion

The baseline experiment with lexical stress obtained from a pronunciation lexicon for prediction of pitch accent yields substantially higher accuracy than chance. This could be particularly useful in resource-limited languages where prosody labels are usually not available but one has access to a reasonable lexicon with lexical stress markers. Off-the-shelf speech synthesizers like Festival and AT&T speech synthesizer perform reasonably well in pitch accent and boundary tone prediction. AT&T speech synthesizer performs better than Festival in pitch accent prediction and the latter performs better in boundary tone prediction. This can be attributed to better rules in the AT&T synthesizer for pitch accent prediction. Boundary tones are usually highly correlated with punctuation and Festival seems to capture this well. However, both these synthesizers generate a high degree of false alarms.

Our syntactic-prosodic maximum entropy model proposed in section 5 outperforms previously reported results on pitch accent and boundary tone classification. Much of the gain comes from the robustness of the maximum entropy modeling in capturing the uncertainty in the classification task. Considering the inter-annotator agreement for ToBI labels is only about 81% for pitch accents and 93% for boundary tones, the maximum entropy framework is able to capture the uncertainty present in manual annotation. The supertag feature offers additional discriminative information over the part-of-speech tags (also as shown by (Hirschberg and Rambow, 2001).

The maximum entropy acoustic-prosodic model discussed in section 6.2 performs reasonably well in isolation. This is a simple method and the quantization resolution can be adjusted based on the amount of data available for training. However, the model does not perform as well when combined with the syntactic features. We conjecture that the generalization provided by the acoustic HMM model is complementary to that provided by the maximum entropy model, resulting in better accuracy when combined together as compared to that of a maxent-based acoustic and syntactic model.

The weighted maximum entropy syntactic-prosodic model and HMM acoustic-prosodic model performs the best in pitch accent and boundary tone classification. The classification accuracies are as good as the inter-annotator agreement for the ToBI labels. Our HMM acoustic-prosodic model is a generative model and does not assume the knowledge of word boundaries in predicting the prosodic labels as in most approaches (Hirschberg, 1993; Wightman and Ostendorf, 1994; Hasegawa-Johnson et al., 2005). This makes it possible to have true parallel prosody prediction during speech recognition. The weighted approach also offers flexibility in prosody labeling for either speech synthesis or speech recognition. While the syntactic-prosodic model would be more discriminative for speech synthesis, the acoustic-prosodic model is more appropriate for speech recognition.

8 Conclusions and Future Work

In this paper, we described a maximum entropy modeling framework for automatic prosody labeling. We presented two schemes for prosody labeling that utilize the acoustic and syntactic information from the input utterance, a maximum entropy model that models the acoustic-syntactic information as a sequence and the other that combines the maximum entropy syntactic-prosodic model and a HMM based acoustic-prosodic model. We also used enriched syntactic information in the form of supertags in addition to POS tags. The supertags provide an improvement in both the pitch accent and boundary tone classification. Especially, in the case where the input utterance is automatically POS tagged (and not hand-corrected), supertags provide a marginal but definite improvement in prosody labeling. The maximum entropy syntactic-prosodic model alone resulted in pitch accent and boundary tone accuracies of 85.2% and 91.5% on training and test sets identical to (Chen et al., 2004). As far as we know, these are the best results on the BU corpus using syntactic information alone and a train-test split that does not contain the same speakers. The acoustic-syntactic maximum entropy model performs better than its syntactic-prosodic counterpart for the boundary tone case but is slightly worse for pitch accent scenario partly due to the approximation involved in quantization. But these results are still better than the baseline results from out-of-the-box speech synthesizers. Finally, our combined maximum entropy syntactic-prosodic model and HMM acoustic-prosodic model performs the best with pitch accent and boundary tone labeling accuracies of 86.0% and 93.1% respectively.

As a continuation of our work, we are incorporating our automatic prosody labeler in a speech-to-speech translation framework. Typically, state-of-the-art speech translation systems have a source language recognizer followed by a machine translation system. The translated text is then synthesized in the target language with prosody predicted from text. In this process, some of the critical prosodic information present in the source data is lost during translation. With reliable prosody labeling in the source language, one can transfer the prosody to the target language (this is feasible for languages with phrase level correspondence). The prosody labels by themselves may or may not improve the translation accuracy but they provide a framework where one can obtain prosody labels in the target language from the speech signal rather than depending on a lexical prosody prediction module in the target language.

Acknowledgements

We would like to thank Vincent Goffin, Stephan Kanthak, Patrick Haffner, Enrico Bocchieri for their support with acoustic modeling tools. We are also thankful to Alistair Conkie, Yeon-Jun Kim, Ann Syrdal and Julia Hirschberg for their help and guidance with the synthesis components and ToBI labeling standard.

References

P. D. Agüero, J. Adell, and A. Bonafonte. 2006. Prosody generation for speech-to-speech translation. In *Proceedings of ICASSP*, Toulouse, France, May.

S. Ananthakrishnan and S. Narayanan. 2005. An automatic prosody recognizer using a coupled multi-stream acoustic model and a syntactic-prosodic language model. In *In Proceedings of ICASSP*, Philadelphia, PA, March.

AT&T Natural Voices speech synthesizer. http://www.naturalvoices.att.com.

S. Bangalore and A. K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2), June.

A. Berger, S. D. Pietra, and V. D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

A. W. Black, P. Taylor, and R. Caley. 1998. The Festival speech synthesis system. http://festvox.org/festival.

J. M. Brenier, D. Cer, and D. Jurafsky. 2005. The detection of emphatic words using acoustic and lexical features. In *In Proceedings of Eurospeech.*

I. Bulyko and M. Ostendorf. 2001. Joint prosody prediction and unit selection for concatenative speech synthesis. In *Proc. of ICASSP*.

K. Chen, M. Hasegawa-Johnson, and A. Cohen. 2004. An automatic prosody labeling system using ANN-based syntactic-prosodic model and GMM-based acoustic-prosodic model. In *Proceedings of ICASSP*.

A. Conkie, G. Riccardi, and R. C. Rose. 1999. Prosody recognition from speech utterances using acoustic and linguistic based models of prosodic events. In *Proc. Eurospeech*, pages 523–526, Budapest, Hungary.

M. Gregory and Y. Altun. 2004. Using conditional random fields to predict pitch accent in conversational speech. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

P. Haffner. 2006. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 48(iv):239–261.

M. Hasegawa-Johnson, K. Chen, J. Cole, S. Borys, S. Kim, A. Cohen, T. Zhang, J. Choi, H. Kim, T. Yoon, and S. Chavara. 2005. Simultaneous recognition of words and prosody in the boston university radio speech corpus. *Speech Communication*, 46:418–439.

J. Hirschberg and C. Nakatani. 1996. A prosodic analysis of discourse segments in direction-giving monologues. In *Proceedings of the 34th conference on Association for Computational Linguistics*, pages 286–293.

J. Hirschberg and O. Rambow. 2001. Learning prosodic features using a tree representation. In *Proceedings of Eurospeech*, pages 1175–1180, Aalborg.

J. Hirschberg. 1993. Pitch accent in context: Predicting intonational prominence from text. *Artificial Intelligence*, 63(1-2).

I. Lehiste. 1970. *Suprasegmentals.* MIT Press, Cambridge, MA.

X. Ma, W. Zhang, Q. Shi, W. Zhu, and L. Shen. 2003. Automatic prosody labeling using both text and acoustic information. In *Proceedings of ICASSP*, volume 1, pages 516–519, April.

E. Nöth, A. Batliner, A. Kießling, R. Kompe, and H. Niemann. 2000. VERBMOBIL: The use of prosody in the linguistic components of a speech understanding system. *IEEE Transactions on Speech and Audio processing*, 8(5):519–532.

M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel. 1995. The Boston University Radio News Corpus. Technical Report ECS-95-001, Boston University, March.

K. Ross and M. Ostendorf. 1996. Prediction of abstract prosodic labels for speech synthesis. *Computer Speech and Language*, 10:155–185, Oct.

P. Shimei and K. McKeown. 1999. Word informativeness and automatic pitch accent modeling. In *In Proceedings of EMNLP/VLC*, College Park, Maryland.

K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. 1992. ToBI: A standard for labeling English prosody. In *Proceedings of ICSLP*, pages 867–870.

X. Sun. 2002. Pitch accent prediction using ensemble machine learning. In *Proc. of ICSLP*.

A. K. Syrdal and J. McGory. 2000. Inter-transcriber reliability of tobi prosodic labeling. In *Proc. ICSLP*, pages 235–238, Beijing, China.

P. Taylor. 1998. The tilt intonation model. In *Proc. ICSLP*, volume 4, pages 1383–1386.

C. W. Wightman and M. Ostendorf. 1994. Automatic labeling of prosodic patterns. *IEEE Transactions on Speech and Audio Processing*, 2(3):469–481.

To Memorize or to Predict: Prominence Labeling in Conversational Speech

A. Nenkova, J. Brenier, A. Kothari, S. Calhoun[†], L. Whitton, D. Beaver, D. Jurafsky
Stanford University
{anenkova,jbrenier,anubha,lwhitton,dib,jurafsky}@stanford.edu
†University of Edinburgh
Sasha.Calhoun@ed.ac.uk

Abstract

The immense prosodic variation of natural conversational speech makes it challenging to predict which words are prosodically *prominent* in this genre. In this paper, we examine a new feature, *accent ratio*, which captures how likely it is that a word will be realized as prominent or not. We compare this feature with traditional accent-prediction features (based on part of speech and N-grams) as well as with several linguistically motivated and manually labeled information structure features, such as whether a word is *given*, *new*, or *contrastive*. Our results show that the linguistic features do not lead to significant improvements, while accent ratio alone can yield prediction performance almost as good as the combination of any other subset of features. Moreover, this feature is useful even across genres; an accent-ratio classifier trained only on conversational speech predicts prominence with high accuracy in broadcast news. Our results suggest that carefully chosen lexicalized features can outperform less fine-grained features.

1 Introduction

Being able to predict the *prominence* or *pitch accent* status of a word in conversational speech is important for implementing text-to-speech in dialog systems, as well as in detection of prosody in conversational speech recognition.

Previous investigations of prominence prediction from text have primarily relied on robust surface features with some deeper information structure features. Surface features like a word's part-of-speech (POS) (Hirschberg, 1993) and its unigram and bigram probability (Pan and McKeown, 1999; Pan and Hirschberg, 2000) are quite useful; content words are much more likely to be accented than function words, and words with higher probability are less likely to be prominent. More sophisticated linguistic features have also been used, generally based on information-structural notions of *contrast*, *focus*, or *given-new*. (Hirschberg, 1993).

For example, in the Switchboard utterance below, there is an intrinsic contrast between the words "women" and "men", making both terms more salient (words in all capital letters represent prominent tokens):

you SEE WOMEN$_c$ GOING off to WARS as WELL as MEN$_c$.

Similarly the givenness of a word may help determine its prominence. The speaker needs to focus the hearer's attention on *new* entities in the discourse, so these are likely to be realized as prominent. *Old* entities, on the other had, need not be prominent; these tendencies can be seen in the following example.

they$_{old}$ have all the WATER$_{new}$ they$_{old}$ WANT. they$_{old}$ can ACTUALLY PUMP water$_{old}$.

While previous models have attempted to capture global properties of words (via POS or unigram probability), they have not in general used word identity as a predictive feature, assuming either that current supervised training sets would be too small or that word identity would not be robust across genres (Pan et al., 2002). In this paper, we show a way to capture word identity in a feature, **accent ratio**, that works well with current small supervised training sets, and is robust to genre differences.

We also use a corpus which has been hand-labeled for information structure features (including given/new and contrast information) to investigate the relative usefulness of both linguistic and shallow features, as well as how well different features combine with each other.

[0]Thanks to the Edinburgh-Stanford Link and ONR (MURI award N000140510388) for generous support.

2 Data and features

For our experiments we use 12 Switchboard (Godfrey et al., 1992) conversations, 14,555 tokens in total. Each word was manually labeled for presence or absence of pitch accent[1] , as well as additional features including information status (or givenness), contrast and animacy distinctions, (Nissim et al., 2004; Calhoun et al., 2005; Zaenen et al., 2004), features that linguistic literature suggests are predictive of prominence (Bolinger, 1961; Chafe, 1976).

All of the features described in detail below have been shown to have statistically significant correlation with prominence (Brenier et al., 2006).

Information status The information status (IS), or givenness, of discourse entities is important for choosing appropriate reference form (Prince, 1992; Gundel et al., 1993) and possibly plays a role in prominence decisions as well (Brown, 1983). No previous studies have examined the usefulness of information status in *naturally occurring* conversational speech. The annotation in our corpus is based on the givenness hierarchy of Prince: first mentions of entities were marked as *new* and subsequent mentions as *old*. Entities that are not previously mentioned, but that are generally known or semantically related to other entities in the preceding context are marked as *med*iated. Obviously, the givenness annotation applies only to referring expressions, i.e. noun phrases the semantic interpretation of which is a discourse entity. This restriction inherently limits the power of the feature for prominence prediction, which has to be performed for all classes of words. Complete details of the IS annotation can be found in (Nissim et al., 2004).

Kontrast One reason speakers make entities in an utterance prominence is because of information structure considerations (Rooth, 1992; Vallduví and Vilkuna, 1998). That is, parts of an utterance which distinguish the information the speaker actually says from the information they could have said, are made salient, e.g. because that information answers a question, or contrasts with a similar entity in the context. Several possible triggers of this sort of salience were marked in the corpus, with words that were not kontrastive (in this sense) being marked as *background*:

- *contrastive* if the word is directly differentiated from a previous topical or semantically-related word;
- *subset* if it refers to a member of a more general set mentioned in the surrounding context;
- *adverbial* if a focus-sensitive adverb such as “only” or “even” is associated with the word being annotated;
- *correction* if the speaker intended to correct or clarify a previous word or phrase;
- *answer* if the word completes a question by the other speaker;
- *nonapplic* for filler phrases such as “in fact”, “I mean”, etc.

Note that only content words in full sentences were marked for kontrast, and filler phrases such as “in fact” and “I mean” were excluded. A complete description of the annotation guidelines can be found in (Calhoun et al., 2005).

Animacy Each noun and pronoun is labeled for the animacy of its referent (Zaenen et al., 2004). The categories include *concrete*, *non-concrete*, *human*, *org*anizations, *place*, and *time*.

Dialog act Specifies the function of the utterance such as *statement*, *opinion*, *agree*, *reject*, *abandon*; or type of question (*yes/no, who, rhetoric*)

In addition to the above theoretically motivated features, we used several automatically derivable word measures.

Part-of-speech Two such features were used, the full Penn Treebank tagset (called POS) , and a collapsed tagset (called BroadPOS) with six broad categories (nouns, verbs, function words, pronouns, adjectives and adverbs).

Unigram and bigram probability These features are defined as $log(p_w)$ and $log(p_{w_i}|p_{w_{i-1}})$ respectively and their values were calculated from the Fisher corpus (Cieri et al., 2004). High probability words are less likely to be prominent.

TF.IDF This measure captures how central a word is for a particular conversation. It is a function of the frequency of occurrence of the word in the conversation (n_w), the number of conversations that contain the word in a background corpus (k) and the number of all conversations in the background corpus (N). Formally, *TF.IDF1* $= n_w \times log(\frac{N}{k})$. We

[1]Of all tokens, 8,429 (or 58%) were not accented.

also used a variant, *TF.IDF2*, computed by normalizing *TF.IDF1* by the number of occurrences of the most frequent word in the conversation. *TF.IDF2* = *TF.IDF1*$/max(n_{w \in conv})$. Words with high *TF.IDF* values are important in the conversation and are more likely to be prominent.

Stopword This is a binary feature indicating if the word appears in a high-frequency stopword list from the Bow toolkit (McCallum, 1996). The list spans both function and content word classes, though numerals and some nouns and verbs were removed.

Utterance length The number of words.

Length The number of characters in the words. This feature is correlated with phonetic features that have been shown to be useful for the task, such as the number of vowels or phones in the word.

Position from end/beginning The position of the word in the utterance divided by the number of words that precede the current word.

Accent ratio This final (new) feature takes the "memorization" of previous productions of a given word to the extreme, measuring how likely it is that a word belongs to a prominence class or not. Our feature extends an earlier feature proposed by (Yuan et al., 2005), which was a direct estimate of how likely it is for the word to be accented as observed in some corpus. (Yuan et al., 2005) showed that the original accent ratio feature was not included in the best set of features for accent prediction. We believe the reason for this is the fact that the original accent ratio feature was computed for all words, even words in which the value was indistinguishable from chance (.50). Our new feature incorporates the significance of the prominence probability, assuming a default value of 0.5 for those words for which there is insufficient evidence in the training data. More specifically,

$$AccentRatio(w) = \begin{cases} \frac{k}{n} & \texttt{if } B(k, n, 0.5) \leq 0.05 \\ 0.5 & \texttt{otherwise} \end{cases}$$

where k is the number of times word w appeared accented in the corpus, n is the total number of times the word w appeared, and $B(k, n, 0.5)$ is the probability (under a binomial distribution) that there are k successes in n trials if the probability of success and failure is equal. Simply put, the accent ratio of a word is equal to the estimated probability of the word being accented if this probability is significantly different from 0.5, and equal to 0.5 otherwise. For example, *AccentRatio(you)=0.3407*, *AccentRatio(education)=0.8666*, and *AccentRatio(probably)=0.5*.

Many of our features for accent prediction are based only on the 12 training conversations. Other features, such as the unigram, bigram, and TF*IDF features, are computed from larger data sources. Accent ratio is also computed over a larger corpus, since the binomial test requires a minimum of six occurrences of a word in the corpus in order to get significance and assign an accent ratio value different from 0.5. We thus used 60 Switchboard conversations (Ostendorf et al., 2001), annotated for pitch accent, to compute k and n for each word.

3 Results

For our experiments we used the J48 decision trees in WEKA (Witten and Frank, 2005). All the results that we report are from 10-fold cross-validation on the 12 Switchboard conversations.

Some previous studies have reported results on prominence prediction in conversational speech with the Switchboard corpus. Unfortunately these studies used different parts of the corpus or different labelings (Gregory and Altun, 2004; Yuan et al., 2005), so our results are not directly comparable. Bearing this difference in mind, the best reported results to our knowledge are those in (Gregory and Altun, 2004), where conditional random fields were used with both textual, acoustic, and oracle boundary features to yield 76.36% accuracy.

Table 1 shows the performance of decision tree classifiers using *a single* feature. The majority class baseline (not accented) has accuracy of 58%. Accent ratio is the most predictive feature: the accent ratio classifier has accuracy of 75.59%, which is two percent net improvement above the previously known best feature (unigram). The accent ratio classifier assigns a "no accent" class to all words with accent ratio lower than 0.38 and "accent" to all other words. In Section 4 we discuss in detail the accent ratio dictionary, but it is worth noting that it does correctly classify even some high-frequency function words like "she", "he", "do" or "up" as accented.

3.1 Combining features

We would expect that a combination of features would lead to better prediction when compared to a classifier based on a single feature. Several past studies have examined classes of features. In order to quantify the utility of different specific features, we ran exhaustive experiments producing classifiers with all possible combinations of two, three, four and five features.

As we can see from figure 1 and table 2, the classifiers using accent ratio as a feature perform best, for all sizes of feature sets. Moreover, the increase of performance compared to a single-feature classifier is very slight when accent ratio is used as feature. Kontrast seems to combine well with accent ratio and all of the best classifiers with more than one feature use kontrast in addition to accent ratio. This indicates that automatic detection of kontrast can potentially help in prominence prediction. But the gains are small, the best classifiers without kontrast but still including accent ratio perform within 0.2 percent of the classifiers that use both.

On the other hand, classifiers that do not use accent ratio perform poorly compared to those that do, and even a classifier using five features (unigram, broad POS, token length, position from beginning and bigram) performs about as well as a classifier using solely accent ratio as a feature. Also, when accent ratio is not used, the overall improvement of the classifier grows faster with the addition of new features. This suggest that accent ratio provides rich information about words beyond that of POS class and general informativeness.[2]

Table 2 gives the specific features in $(n + 1)$-feature classifiers that lead to better results than the best n-classifier. The figures are for the classifiers performing best overall. Interestingly, none of these best classifiers for all feature set sizes uses POS or unigram as a feature. We assume that accent ratio captures all the relevant information that is present in the unigram and POS features. The best classifier with five features uses, in addition to accent ratio, kontrast, tf.idf, information status and distance from the beginning of the utterance. All of these features convey somewhat orthogonal information: semantic, topicality, discourse and phrasing information respectively. Still, all of them in combination improve the performance over accent ratio as a single feature only by one percent.

[2]To verify this we will examine the accent ratio dictionary in closer detail in the next section.

Accent Ratio (AR)	75.59%
AR + Kontrast	76.15%
AR + END/BEG	75.91%
AR + tf.idf2	75.82%
AR + Info Status	75.82%
AR + Length	75.77%
AR + tf.idf1	75.74%
AR + unigram	75.71%
AR + stopword	75.70%
AR + kontrast + length	76.45%
AR + kontrast + BEG	76.24%
AR + kontrast + unigram	76.24%
AR + kontrast + tf.idf1	76.24%
AR + kontrast + length + tfidf1	76.56%
AR + kontrast + length + stopword	76.54%
AR + kontrast + length +tf.idf2	76.52%
AR + kontrast + Status + BEG	76.47%
AR + kontrast + tf.idf1 + Status + BEG	76.65%
AR + kontrast + tf.idf2 + Status + BEG	76.58%

Table 2: Performance increase augmenting the accent ratio classifier.

Figure 1 shows the overall improvement of classifiers with the addition of new features in three scenarios: overall best, best when kontrast is not used as a feature and best with neither kontrast nor accent ratio. The best classifier with five features that do not include kontrast has accent ratio, broad POS, word length, stopword and bigram as features and has accuracy of 76.28%, or just 0.27% worse than the overall best classifier that uses kontrast and information status. This indicates that while there is some benefit to using the two features, they do not lead to any substantial boost in performance. Strikingly, the best classifier that uses neither accent ratio nor kontrast performs very similarly to a classifier using accent ratio as the only feature: 75.82% for the classifier using unigram, POS, tf.idf1, word length and position from end of the utterance.

3.2 The power of linguistic features

One of the objectives of our study was to assess how useful gold-standard annotations for complex linguistic features are for the task of prominence prediction. The results in this section indicate that animacy distinctions (concrete/non-concrete, person, time, etc) and dialog act did not have much power

AccentRatio	unigram	stopword	POS	tf.idf2	tf.idf1	BroadPos	Length	Kontrast	bigram	Info Stat
75.59	73.77	70.77	70.28	70.14	69.50	68.64	67.64	67.57	65.87	64.13

Table 1: Single feature classifi er performance. Features not in the table (position from end, animacy, utterance length and dialog act) all achieve lower accuracy of around 60%

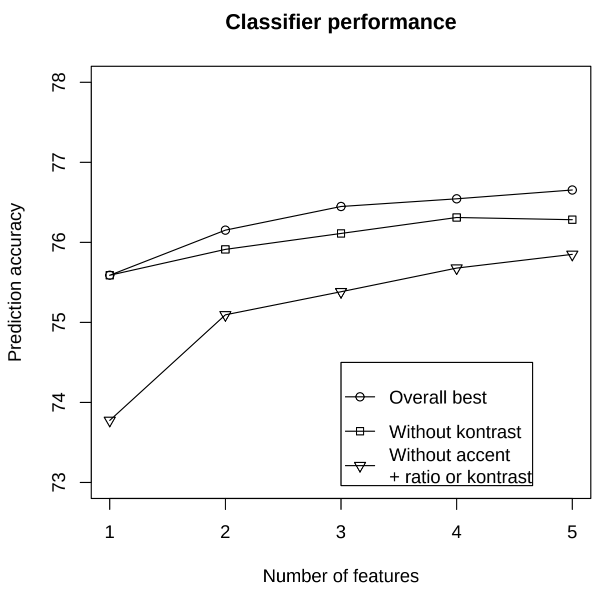


Figure 1: Performance increase with the addition of new features.

as individual features (table 1) and were never included in a model that was best for a given feature set size (table 2).

Information status is somewhat useful and appears in the overall best classifi er with fi ve features (table 2). But when compared with other classifi ers with the same number of features, the benefi ts from adding information status to the model are small. For example, the accent ratio + information status classifi er performs 0.23% better than accent ratio alone, but so does the classifi er using accent ratio and tf.idf. There are two reasons that can explain why the givenness of the referent is not as helpful as we might have hoped. First of all, the information status distinction applies only to referring expressions and has undefi ned values for words such as verbs, adjectives or function words. Second, information status of an entity influences the form of referring expression that is used, with *old* items being more likely to be pronominalized. In the numerous cases where pronominalization of *old* information does occur, features such as POS, unigram or accent ratio will be sensitive to the change of information status simply based on the lexical item.

Kontrast is by far the most useful linguistic feature. It is used in all of the best classifi ers for any feature set size (table 2). It applies to more words than givenness does, since salience distinctions can be made for any part-of-speech class. Still, not all words were annotated for kontrast either, and moreover kontrast only captures one kind of semantic salience. This is particularly true of discourse markers like “especially” or “defi nitely”: these would either be in sentence fragments that weren’t marked for kontrast, or would probably be marked as ’background’ since they are not salience triggers in a semantic sense. As we can see from fi gure 1, classifi ers that use kontrast perform only slightly better than others that use only “cheaper” features.

4 The accent ratio dictionary

Contrary to our initial expectations, both classes in the accent ratio dictionary (for both low and high probability of being prominent) cover the full set of possible POS categories. Tables 3 and 4 list words in both classes (with words sorted by increasing accent ratio in each column). The “no accent” class is dominated by function words, but also includes nouns and verbs. One of the drawbacks of POS as a feature for prominence prediction is that normally auxiliary verbs will be tagged as “VB”, the same class as other more contentful verbs. The informativeness (unigram probability) of a word would distinguish between these types of verbs, but so does the accent ratio measure as well.

Furthermore, some relatively frequent words such as “too”, “now”, “both”, “no”, “yes”, “else”, “wow” have high accent ratio, that is, a high probability for accenting. Such distinctions within the class of function words would not be possible on the basis of in-

.00–.08	.09–.16	.17–.24	.25–.32	.33–.42
a	could	you'd	being	me
uh	in	because	take	i've
um	minutes	oh	said	we're
uh-huh	and	since	wanna	went
the	by	says	been	over
an	who	us	those	you
of	grew	where	into	thing
to	cause	they've	little	what
were	gonna	am	until	some
as	about	sort	they're	out
than	their	you're	I	had
with	but	didn't	that	make
at	on	her	don't	way
for	be	going	this	did
from	through	i'll	should	anything
or	which	will	type	i'm
you've	are	our	we	kind
was	we'll	just	so	go
would	during	though	have	stuff
it	huh	like	got	then
when	is	your	new	she
them	bit	needs	mean	he
it's	there's	my	much	do
if	any	many	i'd	up
can	has	they	know	
him	stayed	get	doesn't	
these	supposed	there	even	

Table 3: Accent ratio entries with low prominence probability.

.58–.74	.75–.79	.80–.86	.87–1.0
lot	both	sometimes	half
time	no	change	topic
now	seems	child	else
kids	life	young	obviously
old	tell	Texas	themselves
too	ready	town	wow
really	easy	room	gosh
three	heard	pay	anyway
work	isn't	interesting	Dallas
nice	again	true	outside
yeah	first	mother	mostly
two	right	problems	yes
person	children	agree	great
day	married	war	exactly
working	may	needed	especially
job	happen	told	definitely
talking	business	finally	lately
usually	still	neat	thirty
rather	daughter	sure	higher
places	gone	house	forty
government	guess	okay	hey
ten	news	seven	Iowa
parents	major	best	poor
paper	fact	also	glad
actually	five	older	basic

Table 4: Accent ratio values for words with high probability for being accented.

formativeness, POS, or even information structure features. Another class like that is words like "yes", "okay", "sure" that are mostly accented by virtue of being the only word in the phrase.

Some rather common words, "not" for example, are not included in the accent ratio dictionary because they do not exhibit a statistically strong preference for a prominence class. The accent ratio classifi er would thus assign class "accented" to the word "not", which is indeed the class this word occurs in more often.

Another fact that becomes apparent with the inspection of the accent ratio dictionary is that while certain words have a statistically signifi cant preference for deaccenting, there is also a lot of variation in their observed realization. For example, personal pronouns such as "I" and "you" have accent ratios near 0.33. This means that every third such pronoun was actually realized as *prominent* by the speaker. In a conversational setting there is an implicit contrast between the two speakers, which could partly explain the phenomenon, but the situations which prompt the speaker to realize the distinction in their speech will be the focus of a future linguistic investigation.

Kontrast is helpful in predicting "accented" class for some generally low ratio words. However, even with its help, production variation in the conversations cannot be fully explained. The following examples from our corpus show low accent ratio words (that, did, and, have, had) that were produced as prominent.

so i did THAT. and then i, you know, i DID that for SIX years. AND then i stayed HOME with my SON.

i HAVE NOT, to be honest, HAD much EXPERIENCE with CHILDREN in that SITUATION.

they're going to HAVE to WORK it OUT to WORKING part TIME.

The examples attest to the presence of variation in production: in the fi rst utterance, for example, we see the words "did", "and" and "that" produced both as prominent and not prominent. Intonational phrasing most probably accounts for some of this variation since it is likely that even words that are typically not prominent will be accented if they occur just before or after a longer pause. We come back to this point in the closing section.

5 Robustness of accent ratio

While accent ratio works well for our data (Table 2), a feature based so strongly on memorizing the status of each word in the training data might lead to problems. One potential problem, suggested by Pan et al. (2002) for lexicalized features in general, is whether a lexical feature like accent ratio might be less robust across genres. Another question is whether our defi nition of accent ratio is better than one that does not use the binomial test: we need to investigate whether these statistical tests indeed improve performance. We focus on these two issues in the next two subsections.

Binomial test cut-off

As discussed above, the original accent ratio feature (Yuan et al., 2005) was based directly on the fraction of accented occurrences in the training set. We might expect such a use of raw frequencies to be problematic. Given what we know about word distributions in text (Baayen, 2001), we would expect about half of the words in a big corpus to appear only once. In an accent ratio dictionary without binomial test cut-off, all such words will have accent ratio of either exactly 1 or 0, but one or even few occurrences of a word would not be enough to determine statistical signifi cance. By contrast, our modifi ed accent ratio feature uses binomial test cut-off to make the accent ratio more robust to small training sets.

To test if the binomial test cut-off really improved the accent ratio feature, we compared the performance on Switchboard of classifi ers using accent ratio with and without cut-off. The binominal test improved the performance of the accent ratio feature from 73.49% (Yuan *et al.* original version) to 75.59% (our version).

Moreover, Yuan *et al.* report that their version of the feature did not combine well with other features, while in our experiments best performance was always achieved by the classifi ers that made use of the accent ratio feature in addition to others.

A cross-genre experiment: broadcast news

In a systematic analysis of the usefulness of different informativeness, syntactic and semantic features for prominence prediction, Pan et al. (2002) showed that *word identity* is a powerful feature. But they hypothesized that this would not be a useful feature in a domain independent pitch accent prediction task. Their hypothesis that word identity cannot be a robust across genres would obviously carry over to accent ratio. In order to test the hypothesis, we used the accent ratio dictionary derived from the Switchboard corpus to predict prominence in the Boston University Radio corpus of broadcast news. Using an accent ratio dictionary from Switchboard and assigning class "not accented" to words with accent ratio less than 0.38 and "accented" otherwise leads to 82% accuracy of prediction for this broadcast news corpus. If the accent ratio dictionary is built from the BU corpus itself, the performance is 83.67%.[3] These results indicate that accent ratio is a robust enough feature and is applicable across genres.

6 Conclusions and future work

In this paper we introduced a new feature for prominence prediction, accent ratio. The accent ratio of a word is the (maximum likelihood estimate) probability that a word is accented if there is a signifi - cant preference for a class, and 0.5 otherwise. Our experiments demonstrate that the feature is powerful both by itself and in combination with other features. Moreover, the feature is robust to genre, and accent ratio dictionaries can be used for prediction of prominence in read news with very good results.

Of the linguistic features we examined, kontrast is the only one that is helpful beyond what can be gained using shallow features such as n-gram probability, POS or tf.idf. While the improvements from kontrast are relatively small, the consistency of these small improvements suggest that developing automatic methods for approximating the gold-standard annotation we used here, similar to what has been done for information status in (Nissim, 2006), may be worthwhile. An automatic predictor for kontrast may also be helpful in other applications such as question answering or textual entailment.

All of the features in our study were text-based. There is a wide variety of research investigating phonological or acoustic features as well. For example Gregory and Altun (2004) used acoustic features

[3]This result is comparable with the result of (Yuan et al., 2005) who in their experiment with the same corpus report the best result as 83.9% using *three* features: unigram, bigram and backwards bigram probability.

such as duration and energy, and phonological features such as oracle (hand-labeled) intonation phrase boundaries, and the number of phones and syllables in a word. Although acoustic features are not available in a text-to-speech scenario, we hypothesize that in a task where such features are available (such as in speech recognition applications), acoustic or phonological features could improve the performance of our text-only features. To test this hypothesis, we augmented our best 5-feature classifier which did not include kontrast with hand-labeled intonation phrase boundary information. The resulting classifier reached an accuracy of 77.45%, more than one percent net improvement over 76.28% accuracy of the model based solely on text features and not including kontrast. Thus in future work we plan to incorporate more acoustic and phonological features.

Finally, prominence prediction classifiers need to be incorporated in a speech synthesis system and their performance should be gauged via listening experiments that test whether the incorporation of prominence leads to improvement in synthesis.

References

R. H. Baayen. 2001. *Word Frequency Distributions*. Kluwer Academic Publishers.

D.L. Bolinger. 1961. Contrastive Accent and Contrastive Stress. *Language*, 37(1):83–96.

J. Brenier, A. Nenkova, A. Kothari, L. Whitton, D. Beaver, and D. Jurafsky. 2006. The (non)utility of linguistic features for predicting prominence in spontaneous speech. In *IEEE/ACL 2006 Workshop on Spoken Language Technology*.

G. Brown. 1983. Prosodic structure and the given/new distinction. *Prosody: Models and Measurements*, pages 67–77.

S. Calhoun, M. Nissim, M. Steedman, and J.M. Brenier. 2005. A framework for annotating information structure in discourse. *Pie in the Sky: Proceedings of the workshop, ACL*, pages 45–52.

W. Chafe. 1976. Givenness, contrastiveness, definiteness, subjects, topics, and point of view. *Subject and Topic*, pages 25–55.

C. Cieri, D. Graff, O. Kimball, D. Miller, and Kevin Walker. 2004. Fisher English training speech part 1 transcripts. *LDC*.

J. Godfrey, E. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *IEEE ICASSP-92*.

M. Gregory and Y. Altun. 2004. Using conditional random fields to predict pitch accents in conversational speech. *Proceedings of ACL*, 2004.

J. Gundel, N. Hedberg, and R. Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

J. Hirschberg. 1993. Pitch Accent in Context: Predicting Intonational Prominence from Text. *Artificial Intelligence*, 63(1-2):305–340.

A. McCallum. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow.

M. Nissim, S. Dingare, J. Carletta, and M. Steedman. 2004. An annotation scheme for information status in dialogue. In *LREC 2004*.

M. Nissim. 2006. Learning information status of discourse entities. In *Proceedings of EMNLP 2006*.

M. Ostendorf, I. Shafran, S. Shattuck-Hufnagel, L. Carmichael, and W. Byrne. 2001. A prosodically labeled database of spontaneous speech. *Proc. of the ISCA Workshop on Prosody in Speech Recognition and Understanding*, pages 119–121.

S. Pan and J. Hirschberg. 2000. Modeling local context for pitch accent prediction. In *Proceedings of ACL-00*.

S. Pan and K. McKeown. 1999. Word informativeness and automatic pitch accent modeling. In *Proceedings of EMNLP/VLC-99*.

S. Pan, K. McKeown, and J. Hirschberg. 2002. Exploring features from natural language generation in prosody modeling. *Computer speech and language*, 16:457–490.

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.

Mats Rooth. 1992. A theory of focus interpretation. *Natural Language Semantics*, 1(1):75–116.

E. Vallduví and M. Vilkuna. 1998. On rheme and kontrast. *Syntax and Semantics*, 29:79–108.

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

J. Yuan, J. Brenier, and D. Jurafsky. 2005. Pitch Accent Prediction: Effects of Genre and Speaker. *Proceedings of Interspeech*.

A. Zaenen, J. Carletta, G. Garretson, J. Bresnan, A. Koontz-Garboden, T. Nikitina, M.C. O'Connor, and T. Wasow. 2004. Animacy Encoding in English: why and how. *ACL Workshop on Discourse Annotation*.

Avoiding and Resolving Initiative Conflicts in Dialogue*

Fan Yang and **Peter A. Heeman**
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
{fly,heeman}@cslu.ogi.edu

Abstract

In this paper, we report on an empirical study on initiative conflicts in human-human conversation. We examined these conflicts in two corpora of task-oriented dialogues. The results show that conversants try to avoid initiative conflicts, but when these conflicts occur, they are efficiently resolved by linguistic devices, such as volume.

1 Introduction

Current computer dialogue systems tend to be system-initiative. Although there are some mixed-initiative systems that allow the user to make a request or state a goal, such systems are limited in how they follow natural initiative behavior. An example is where the system always releases the turn whenever the user barges in. However, in a complex domain where the computer system and human user are collaborating on a task, the computer system might need to interrupt the human user, or might even need to fight with the human user over the turn. Thus the next generation of computer dialogue systems need a better model of initiative (Horvitz, 1999). In what situations can the system try to take initiative from the user? What devices can the system use to fight for initiative? We propose examining human-human conversation to answer these questions. Once we understand the conventions people adopt in negotiating initiative, we can implement them in a computer dialogue system to create natural interactivity.

In this research work, we examined two corpora of human-human conversation: the Trains corpus (Heeman and Allen, 1995) and the MTD corpus (Heeman et al., 2005). The research purpose is to understand conversants' behavior with initiative conflicts, which we define a situation where both conversants try to direct the conversation at the same time, but one of them fails. We found that (1) conversants try to avoid initiative conflicts; and (2) initiative conflicts, when they occur, are efficiently resolved by linguistic devices, such as volume.

In Section 2, we review related research work on modeling initiative and turn-taking. Dialogue initiative and turn-taking are two intertwined research topics. When conversants fight to show initiative, they are also fighting for the turn to speak. In Section 3, we describe the two corpora and their annotations. In Section 4, we define initiative conflict and give an example. In Section 5, we present the evidence that conversants try to avoid initiative conflicts. In Section 6, we present evidence that initiative conflicts are efficiently resolved by linguistic devices. We discuss our findings in Section 7 and future work in Section 8.

2 Related Research

2.1 Initiative Models

Researchers have been investigating how people manage dialogue initiative in their conversation. Whittaker and Stenton (1988) proposed rules for tracking initiative based on utterance types; for example, statements, proposals, and questions show initiative, while answers and acknowledgements do not. Smith (1993) proposed four different initiative strategies with differing amounts of control by the system. Chu-Carrol and Brown (1998) distinguished dialogue initiative from task initiative, and proposed an evidential model of tracking both of them. Cohen et al. (1998) proposed presenting initiative in different strengths. Some researchers related initiative to discourse structure. Walker and Whittaker (1990) found a correlation between initiative switches and discourse segments. Strayer et al. (2003) proposed the *restricted initiative model* in which the initiator of a discourse segment, who introduces the discourse segment purpose, is in control of the segment and shows most of the initiative. These models allowed the possibility that multiple conversants will want to show initiative at the same time; however, none of them addressed initiative conflicts.

Guinn (1998) studied another type of initiative, task initiative, which is about directing the problem-solving

*This work was funded by the National Science Foundation under IIS-0326496.

of a domain goal. Guinn proposed that the person who is more capable of coordinating the current goal is the person who should be leading the dialogue. Initiative switches between conversants as goals get pushed and popped from the problem-solving stack. However, because conversants only have incomplete information, initiative conflicts might occur when conversants overestimate their own capability or underestimate the other's. Guinn proposed a negotiation model to resolve these conflicts of task initiative. Conversants negotiate by informing each other of positive and negative information of their plans to achieve the goal. By comparing each other's plan, the conversant whose plan has the higher probability of success takes initiative. Guinn's research on conflicts of task initiative, however, has little bearing on conflicts of dialogue initiative. For dialogue initiative, very often, one of the conversants just gives up the attempt very quickly, without giving a justification. As stated by Haller and Fossum (1999):"... conflicts are often simple clashes that result from both participants trying to take the initiative at the same time. Such conflicts do not necessarily require complex negotiation to resolve. Often, unwritten rules based on factors like social roles, personal assertiveness, and the current locus of control play a part in determining who will give away." However, Haller and Fossum did not further investigate how conversants efficiently resolve conflicts of dialogue initiative.

2.2 Turn-Taking and Initiative

Turn-taking in conversation is highly related to initiative. Conversants have to possess the turn in order to show initiative. When conversants are fighting for initiative, they are also fighting for the turn to speak. Thus the mechanisms of turn-taking might share some similarity with initiative. On the other hand, turn-taking is different from initiative; for example, an answer takes a turn, but answering does not show initiative.

Turn-taking in conversation has been discussed in linguistics literature. Duncan (1974) examined cues (gesture, acoustic, and linguistic) that conversants use to signal turn-taking or turn-releasing. A model based on these signals was created to account for conversants' turn-taking behavior. In this model, miscues are the cause of overlapping speech: for example, the hearer misrecognizes the speaker's cue to keep the turn, or the speaker fails to properly signal.

Sacks et al. (1974) proposed a set of rules for turn-taking: the current speaker can select somebody else to speak; otherwise, hearers can self-select to speak; otherwise, the speaker can self-select to speak. This model suggested that overlapping speech results from either the hearer waiting too long to speak, or the speaker not waiting long enough.

Schegloff (2000) examined overlapping speech in detail in human conversation. He concluded that (1) fights for turn are often accompanied with sudden acoustic alteration, such as louder volume, higher pitch, and faster or slower speaking rate; (2) the vast majority of fights for turn are resolved very quickly; (3) fights for turn are resolved through an interactive procedure, e.g. syllable by syllable negotiation, using devices such as volume, pitch, and speaking rate. However, his analysis only consisted of a few examples; no statistical evidence was given. It is thus unclear whether his conclusions represent human conventions of initiative conflict, or are occasional behavior that would only occur under special circumstances.

3 Corpora and Annotations

To understand human behavior in initiative conflicts, we examined two corpora, the Trains corpus and the MTD corpus. These two corpora have very different domain setups. The distinct behavior seen in each corpus will help inform us how domain settings affect initiative, while the common behavior will help inform us the cross-domain human conventions.

3.1 The Trains Corpus

The Trains corpus is a collection of human-human task-oriented dialogues, in which two participants work together to formulate a plan involving the manufacture and transportation of goods. One participant, the user, has a goal to solve; and the other participant, the system, knows the detailed domain information including how long it takes to ship and manufacture goods.

We annotated eight Trains dialogues totaling about 45 minutes using the tool DialogueView (Yang et al., 2007). We tagged each utterance with a simplified DAMSL scheme (Core and Allen, 1997). Utterances were tagged as forward or backward functions, stalls, or non-contributions. Forward functions include statements, questions, checks and suggestions. Backward functions include agreements, answers, acknowledgments, repetitions and completions. Examples of stalls are "um" and "let's see", used by a conversant to signal uncertainty of what to say next or how to say it. Non-contributions include abandoned and ignored utterances. The flow of the dialog would not change if non-contributions were removed.

Hierarchical discourse structure was annotated following Strayer et al. (2003). To determine whether a group of utterances form a discourse segment, we took into account whether there exists a shared goal introduced by one of the conversants (cf. Grosz and Sidner, 1986).

3.2 The MTD Corpus

The MTD corpus contains dialogues in which a pair of participants play two games via conversation: an ongoing game that takes a relatively long time to finish and an interruption game that can be done in a couple turns but has a time constraint. Both games are done on computers. Players are separated so that they cannot see each other.

In the ongoing game, the two players work together to assemble a poker hand of a full house, flush, straight, or four of a kind. Each player has three cards in hand, which the other cannot see. Players take turns drawing an extra card and then discarding one until they find a poker hand, for which they earn 50 points. To discourage players from simply rifling through the cards to look for a specific card without talking, one point is deducted for each picked-up card, and ten points for a missed or incorrect poker hand. To complete this game, players converse to share card information, and explore and establish strategies based on the combined cards in their hands.

From time to time, the computer generates a prompt for one player to start an interruption game to find out whether the other player has a certain picture on the screen. The interruption game has a time constraint of 10, 25, or 40 seconds, which is (pseudo) randomly determined. Players get five points for the interruption game if the correct answer is given in time. Players are told to earn as many points as possible.

We annotated six MTD dialogues totaling about 90 minutes. Utterances were segmented based on player's intention so that each utterance has only one *dialogue act* that is to share information, explore strategies, suggest strategies, or maintain an established strategy (Toh et al., 2006). We applied the same simplified DAMSL scheme on utterance tag annotations. Figure 1 shows an annotated excerpt of an MTD dialogue. We grouped utterances into blocks. Block *b21* is a game block in which conversants completed a poker hand. Blocks *b22* and *b23* are two card blocks in which conversants picked up a new card, discussed what they had in hand, and chose a card to discard. Block *b24* is an interruption segment in which conversants switched their conversation to the interruption game. No claim is made that the game and card blocks are discourse segments according to Grosz and Sidner's definition (1986).

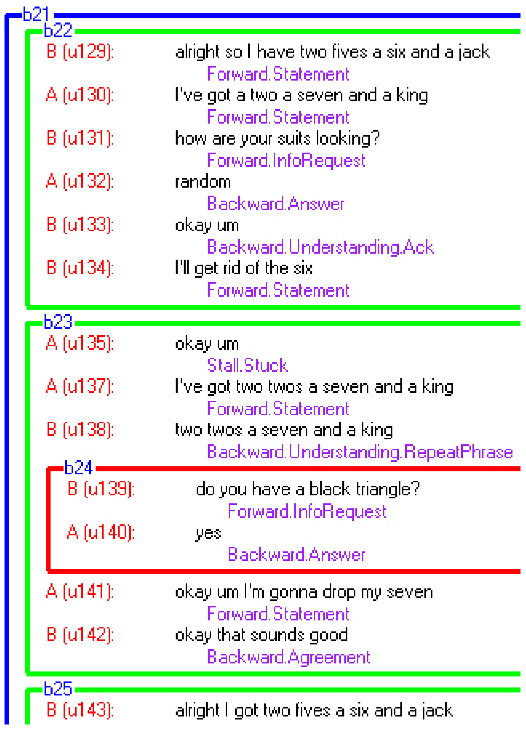

b21
b22
B [u129]: alright so I have two fives a six and a jack
Forward.Statement
A [u130]: I've got a two a seven and a king
Forward.Statement
B [u131]: how are your suits looking?
Forward.InfoRequest
A [u132]: random
Backward.Answer
B [u133]: okay um
Backward.Understanding.Ack
B [u134]: I'll get rid of the six
Forward.Statement
b23
A [u135]: okay um
Stall.Stuck
A [u137]: I've got two twos a seven and a king
Forward.Statement
B [u138]: two twos a seven and a king
Backward.Understanding.RepeatPhrase
b24
B [u139]: do you have a black triangle?
Forward.InfoRequest
A [u140]: yes
Backward.Answer
A [u141]: okay um I'm gonna drop my seven
Forward.Statement
B [u142]: okay that sounds good
Backward.Agreement
b25
B [u143]: alright I got two fives a six and a jack

Figure 1: An excerpt of an MTD dialogue

4 Defining Initiative Conflicts

An initiative conflict occurs when a conversant's attempt to show initiative fails because someone else is showing initiative at the same time. Following Whittaker and Stenton (1988), we use utterance tags to determine whether an utterance shows initiative: forward functions show initiative while others do not. Non-contributions are viewed as failed attempt to show initiative. Thus we identify initiative conflicts as overlapping utterances that involve either a forward function and a non-contribution or two non-contributions.

Figure 2 gives an example of an initiative conflict from the MTD corpus. The top conversant says "that's pair of threes and pair of fours", which ends at time point A. After a short pause, at time B, the bottom conversant asks "how many threes do you have", which is overlapped by the top conversant's second utterance "I'll drop" at time C. The top conversant then abandons the attempt of showing initiative at time D. Hence the bottom speaker is the winner of this initiative conflict.

We use the term *preceding-pause* to refer to the time interval between the end of the previous utterance and the first utterance that is involved in the overlap (from A to B in Figure 2). *Offset* refers to the interval between the start times of the two overlapped utterances (from B to C). *Duration* refers to the time interval from the beginning of overlap till the end of overlap (from C to D).

In the Trains corpus, there are 142 cases of overlapping speech, 28 of which are initiative conflicts. Of the remaining, 96 cases involve a backward function (e.g. an acknowledgment overlapping the end of an inform), and 10 cases involve a stall. The remaining 8 cases are other types of overlap, such as a collaborative completion, or conversants talking about the same thing: for example, one saying "we are a bit early" and the other saying "we are a little better".

In the MTD corpus, there are 383 cases of overlapping speech, 103 of which are initiative conflicts. Of the remaining, 182 cases involve a backward function, 21 cases involve a stall, and 77 cases are others. Initiative conflicts

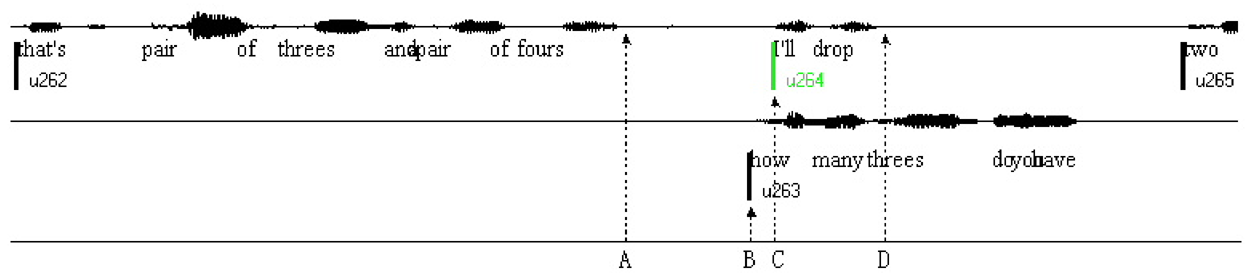

that's pair of threes and pair of fours
u262
I'll drop
u264
two
u265
how many threes do you have
u263
A
B
C
D

Figure 2: An illustration of an initiative conflict

are more frequent in the MTD corpus (103 cases in 90 min) than in the Trains corpus (28 cases in 45 min).

There are three cases in the Trains and thirteen cases in the MTD corpus where the preceding-pause is negative, i.e. the first overlapped utterance is started before the other conversant finishes the previous utterance. Sometimes the hearer starts a little bit early to take the turn. If the original speaker does not intend to release the turn, a conflict arises. Because these cases involve three utterances, we exclude them from our current analysis and save them for future research.[1] This leaves 25 cases in the Trains corpus and 90 cases in the MTD corpus for analyzing initiative conflicts.

5 Avoiding Initiative Conflicts

In this section, we show that conversants try to avoid initiative conflicts by examining both the offset of initiative conflicts and the urgency levels.

5.1 Offset of Initiative Conflicts

The offset of an initiative conflict indicates where the conflict happens. A short offset indicates that the conflict happens at the beginning of an utterance, while a long offset indicates an interruption in the middle.

Figure 3 shows the cumulative distribution function (CDF) for offsets for both corpora individually. The mean offset is 138ms for the Trains corpus, and 236ms for the MTD corpus. In comparison to the average length of forward utterances (2596ms in the Trains corpus and 1614ms in the MTD corpus), the offset is short. Moreover, in the Trains corpus, 88% of offsets are less than 300ms (and 80% less than 200ms); in the MTD corpus, 75% of offsets are less than 300ms. Thus most initiative conflicts happen at the beginning of utterances.

[1]These cases of negative value preceding-pause are in fact very interesting. They seem to contradict with Sacks et al. (1974)'s model that the hearer has priority to self select to speak. If Sacks et al. is correct, the speaker should wait a certain amount of time in order not to overlap with the hearer, but in these cases we see that the speaker self-selects to speak without taking into account whether the hearer self-selects to speak or not.

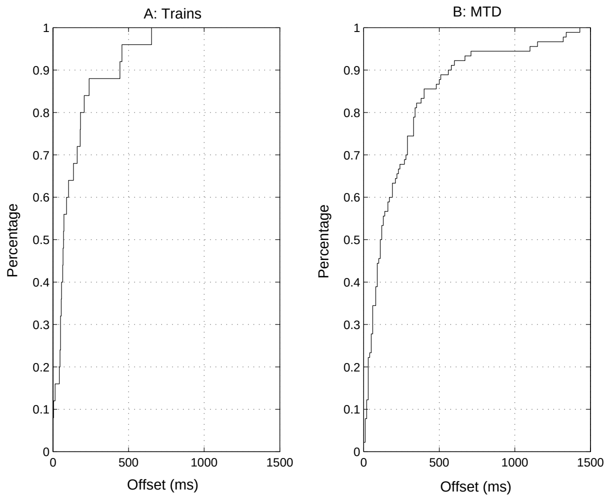


Figure 3: CDF plot for offsets of initiative conflicts

Few initiative conflicts have offsets longer than 500ms. There is one instance in the Trains corpus and eleven in the MTD corpus. Four cases are because the second conversant has something urgent to say. For example, when an interruption game is timing out, conversants would interrupt, sometimes in the middle of an utterance, which results in a long offset. Another six cases are due to miscues. Figure 4 shows an example. Conversant B said "I have two aces" with end-of-utterance intonation, paused for about half a second, and then added "and a seven". The ending intonation and the pause probably misled conversant A to believe that B had finished, and thus A started a new forward utterance, which overlapped with B's extension. A's utterance was then quickly abandoned. In these cases, it is ambiguous whether B's utterance "I have two aces ... and a seven" should be further chopped into two utterances. The final two cases are intrusions, with an example shown in Figure 5. Conversant A cut in probably because he was confident with his decision and wanted to move on to the next card. In such cases, the intruder might be perceived as being rude.

B:	I have two aces	and a seven	
A:		I have	.

Figure 4: Long offset: miscue

B:	well let's just	
A:		it's no help I think it goes away

Figure 5: Long offset: intrusion

The preponderance of short offsets provides evidence that conversants try to avoid initiative conflicts. When A detects that B is talking, A should not attempt to show initiative until the end of B's utterance in order to avoid conflicts, unless there is an urgent reason. If conversants did not take into account whether someone else is speaking before attempting initiative, we would see a lot of intrusions in the middle of utterances, which in fact rarely happen in the two corpora. As we have shown, initiative conflicts tend to happen at the beginning of utterances. Thus initiative conflicts occur mainly due to unintentional collision, i.e. both conversants happen to start speaking almost at the same time. The fact that the offset of most initiative conflicts is within 300ms confirms this.[2]

5.2 Urgency Level and Initiative Conflicts

To further support the hypothesis that conversants avoid initiative conflicts except for urgent reasons, we examined the MTD corpus for the correlation between the urgency levels of the interruption game and initiative conflicts. For the urgency level of 10 seconds, conversants started 33 interruption games, 8 of which were introduced via initiative conflicts. For 25 seconds, conversants started 36 interruption games, 5 introduced via initiative conflicts. For 40 seconds, conversants started 33 interruption games, 3 introduced via initiative conflicts. Thus the percentages of initiative conflicts for the three urgency levels are 24% for 10 seconds, 14% for 25 seconds, and 9% for 40 seconds. The urgency level of 10 seconds requires conversants to start the interruption game very quickly in order to complete it in time. On the other hand, the urgency level of 40 seconds allows conversants ample time to wait for the best time to start the game (Heeman et al., 2005). Thus we see the percentage of initiative conflicts decreases as it becomes less urgent to the interruption game. These results suggest that conversants try to avoid initiative conflicts if they can, unless there is an urgent reason.

6 Resolving Initiative Conflicts

In this section, we present evidence that initiative conflicts, if they occur, are resolved very quickly using simple devices.

[2]This 300ms might be related to human reaction time.

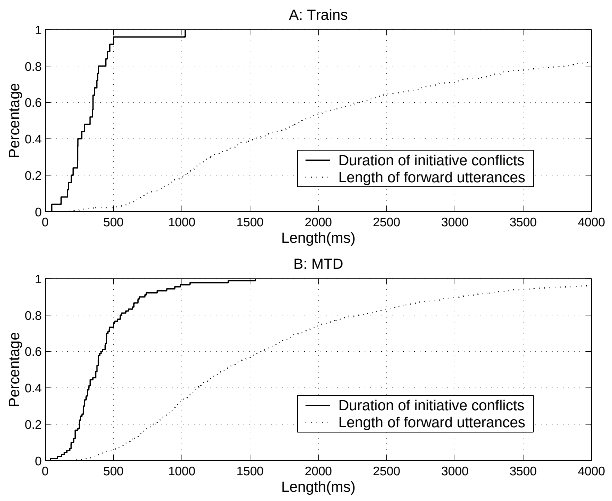


Figure 6: CDF plot for durations of initiative conflicts together with lengths of forward utterances

6.1 Duration of Initiative Conflicts

The duration of an initiative conflict, as defined in Section 4, indicates how quickly the conflict is resolved. Figure 6 shows the cumulative distribution function of durations of initiative conflicts and the lengths of forward utterances in the two corpora. The mean duration is 328ms in the Trains corpus and 427ms in the MTD corpus. From Figure 6 we see that the duration is much shorter than the length of forward utterances, which have the mean length of 2596ms in the Trains corpus and 1614ms in the MTD corpus. The difference between duration of initiative conflicts and length of forward utterances is statistically significant ($p < 10^{-5}$, ttest). On average, the duration of initiative conflicts is about 1/8 the length of forward utterances in the Trains corpus and about 1/4 in the MTD corpus. The short durations suggest that initiative conflicts are resolved very quickly.

According to Crystal and House (1990), the average length of CVC syllable is about 250ms. Thus on average, the length of initiative conflicts is about one to two syllables.[3] In fact, 96% of conflicts in the Trains corpus and 73% in the MTD corpus are resolved within 500ms. These observations are consistent with one of Schelogff's (2000) claims about turn-taking conflicts, that they usually last less than two syllables to resolve.

6.2 Resolution of Initiative Conflicts

From our definition of initiative conflict, at least one of the speakers has to back off. For expository ease, we re-

[3]It would be interesting to examine the length of initiative conflicts based on syllable. However currently we do not have syllable-level alignment for the two corpora. We leave this for future research.

fer to the person who gets the turn to contribute as *the winner*, and the other who fails as *the yielder*. There are two cases in the Trains corpus and three cases in the MTD corpus in which both speakers abandoned their incomplete utterances, paused for a while, and then one of them resumed talking. These five cases are treated as ties: no winners or yielders, and are excluded from our analysis here.

Given how quickly initiative conflicts are resolved, we examined whether the resolution process might be dependent on factors presented before the conflict even begins, namely who was speaker in the previous utterance, and who was interrupted. If we predict that the conversant who spoke prior to the conflict (speaker of u262 in Figure 2) loses, we get 55% accuracy in the Trains corpus and 61% accuracy in the MTD corpus. If we predict the conversant who spoke first in the overlap (speaker of u263 in Figure 2) wins, we get 60% accuracy in the Trains corpus and 53% accuracy in the MTD corpus. These low percentages suggest that they are not robust predictors.

We next examined how conversants resolve the conflicts using devices such as volume, pitch, and others.

6.2.1 Volume

For a stretch of speech, volume is calculated as the mean energy of the spoken words. For each initiative conflict, we calculated each conversant's volume during the overlap, and then normalized it with respect to the conversant's volume throughout the whole conversation.[4] We refer to this as *relative volume*. In the Trains corpus, the average relative volume of the winner is 1.06; the average relative volume of the yielder is 0.93. The difference is statistically significant ($P < 0.01$, anova). In the MTD corpus, the average relative volume of the winner is 1.12; the average relative volume of the yielder is 0.98. The difference is also statistically significant ($p < 10^{-6}$, anova). These results show that the winner is the one speaking at a higher relative volume.

To strengthen our argument, we also calculated *volume ratio* as the relative volume of the winner divided by the yielder. The average volume ratio in the Trains corpus is 1.16 and in the MTD corpus is 1.18. If a classifier always chooses the speaker with higher relative volume to be the winner, we achieve about 79% accuracy in both corpora, which is a 29% absolute improvement over random prediction. These results further confirm that the conversant who speaks at a higher relative volume wins the initiative conflicts.

Given the importance of volume in the resolution process, we examined whether it has an impact on the duration of initiative conflicts. Figure 7 plots the relation

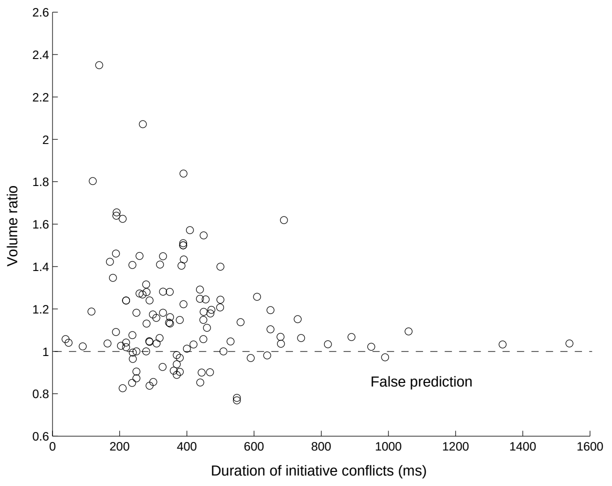


Figure 7: Volume ratio and duration of conflicts

between volume ratio and duration of conflicts for all the cases in the two corpora. For reference, the dotted line divides the data points into two groups: under the line are what volume ratio fails to predict the winner, and above the line are success. If we look at the points where volume ratio succeeds, we see that when duration of initiative conflicts is long, volume ratio tends to be small: in fact, the average volume ratio for initiative conflicts shorter than 600ms is 1.27; for long than 600ms is 1.13; and the difference is statistically significant (ttest, $p < 0.01$).

To further understand how volume is used in the resolution procedure, we examined how volume changes during the overlap. For initiative conflicts whose duration is longer than 600ms, we cut the overlapped speech evenly in half, and calculated the relative volume for each half individually. For the first half, the average relative volume of the winner is 1.03, and the yielder is 1.02. The difference is not statistically significant ($p = 0.93$, paired ttest). For the second half, the average relative volume of the winner is 1.20, and the yielder is 1.02. The difference is statistically significant ($p < 0.001$, paired ttest). The fact that these long initiative conflicts are not resolved in the first half is probably partially due to the close relative volume.

We then calculated *volume increment* as subtracting the relative volume of the first half from the second half. The average volume increment of the winner is 0.17; the average volume increment of the yielder is 0. The difference is statistically significant ($p < 0.001$, paired ttest). These results show that the range of volume increment during the overlap by the winner is larger than the yielder. The behavior of increasing volume during overlap to win the fight suggests that conversants use volume as a device to resolve initiative conflicts.

[4]Normalization is necessary particularly as conversants heard each other via headsets, and the microphones were not calibrated to have exactly the same gains.

6.2.2 Pitch

We used the tool *WaveSurfer* (Sjölander and Beskow, 2000) to extract the f0 from the audio files. We calculated *relative pitch* similarly as we did for volume.

In the Trains corpus, the average relative pitch of the winner is 1.02; the average relative pitch of the yielder is 0.96. The difference is not statistically significant ($P = 0.54$, anova). In the MTD corpus, the average relative pitch of the winner is 1.09; the average relative pitch of the yielder is 0.98. The difference is statistically significant ($p < 0.001$, anova). If we choose the speaker with higher pitch to be the winner, we achieve about 65% accuracy in the Trains corpus and 62% in the MTD corpus. These results suggest that pitch alone is not robust for predicting the winner of initiative conflicts, at least not as predictive as volume, although we do see the tendency of higher pitch by the winner.

We also examined pitch range in the window of 100ms and 300ms respectively. We calculated the pitch range of the overlapping speech and then normalized it with respect to the conversant's pitch range throughout the whole conversation. We did not see a significant correlation between pitch range and the winner of initiative conflicts. Thus pitch does not seem to be a device for resolving initiative conflicts.

6.2.3 Role of Conversants

Human-computer dialogues often have a user interacting with a system, in which the two have very different roles. Hence, we investigated whether the conversant's role has an effect in how initiative conflicts are resolved. We focused on the Trains corpus due to both its rich discourse structure and the difference in the roles that the system and the use have.

In the Trains corpus, if we predict that the initiator of a discourse segment wins the conflicts, we get 65% accuracy. In system-initiated segments, the system wins all eight conflicts; however, in user-initiated segments, the user only wins seven and system wins eight. The user does not have an advantage during initiative conflicts in its segments. Moreover, if the initiator had an advantage, we would expect the system to have fought more strongly in the user-initiated segments in order to win. However, we do not see that the relative volume of the system winning in user-initiated segments is statistically higher than in system-initiated segments in this small sample size ($p = 0.9$, ttest). The initiator does not seem to have a privileged role in the resolution process.

From the above analysis, we see that the system wins the conflicts 16 out of 23 times. Thus if we predict that the system always wins the conflicts, we achieve 70% accuracy. This is not surprising because the system has all the domain information, and is more experienced in solving goals. If the system and user want to speak at the same time, both would know that the system probably has a more significant contribution. That the system wins most of the initiative conflicts agrees with Guinn (1998) that capability plays an important role in determining who to show initiative next.

7 Discussion

In this paper, we present our empirical study of human behavior in initiative conflicts. Our first finding is that conversants try to avoid initiative conflicts. The consequence of initiative conflicts is that at least one of the conversants would have to back off, which makes their effort of contributing in vain. Moreover, the effort of resolving initiative conflicts is overhead to the dialogue. According to the theory of *least collaborative effort* by Clark and Wilkes-Gibbs (1986), it only makes sense for conversants to interrupt when the loss of not interrupting is higher than the cost of an initiative conflict. Thus the theory of least collaborative effort is consistent with our conclusion that most initiative conflicts are unintentional collisions, except where conversants interrupt in the middle of an utterance for urgency reasons.

The second finding of our research is that initiative conflicts, when they occur, are efficiently resolved. We found that volume plays an important role: the louder speaker wins. We also show how conversants change their volume to resolve initiative conflicts. Conversants probably identify their eagerness of speaking, confidence in what they want to say, and capability of achieving the current goal by means of volume, which resolves the initiative conflicts very quickly.

Domain settings obviously have an impact on conversants' initiative behavior. There are more frequent initiative conflicts in the MTD corpus than in the Trains corpus. Moreover, the roles of the conversants also affect their initiative behavior as we found that the system wins more initiative conflicts in the Trains corpus. In a teacher-student conversation, one would expect to see that the teacher interrupts the student more often than vice versa, but also that the teacher wins more initiative conflicts. Capability, culture, and social relationship probably are some underlying elements that influence when and under what conditions conversants would seek initiative, while volume is a device for resolving initiative conflicts.

8 Future Work

In this paper we focused on initiative conflicts in dialogue where two conversants cannot see each other. In face-to-face conversation, there might be other cues, such as eye-contact, head-nodding, and hand gesture, that conversants use in initiative conflicts. Moreover, in a multi-party conversation, a conversant might talk to different people on different topics, and get interrupted from time to time, which leads to an initiative conflict involving multiple speakers. In our future work, we plan to examine initiative conflicts in face-to-face multi-party conversation, such as the ICSI corpus (Shriberg et al., 2004).

Inspired by the findings on human behavior of initiative conflicts, we speculate that conversants might also have a mechanism to even minimize unintentional initiative conflicts, which probably includes devices such as volume, pause, and other prosodic features. The speaker uses these devices, as opposed to explicitly informing each other of their knowledge to evaluate capability (Guinn, 1998), to implicitly signal his or her eagerness, confidence and capability. The hearer then compares his or her own eagerness with the speaker's, and decides whether to just make an acknowledgement (allowing the speaker to continue the lead) or to take over the initiative when taking the turn to speak. In our future work, we plan to build an initiative model to capture this negotiation process.

References

Jennifer Chu-Carroll and Michael K. Brown. 1998. An evidential model for tracking initiative in collaborative dialogue interactions. *User Modeling and User Adapted Interaction*, 8:215–253.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognitive Science*, 22:1–39.

Robin Cohen, C. Allaby, C. Cumbaa, M. Fitzgerald, K. Ho, B. Hui, C. Latulipe, F. Lu, N. Moussa, D. Pooley, A. Qian, and S. Siddiqi. 1998. What is initiative? *User Modeling and User Adapted Interaction*, 8:171–214.

Mark G. Core and James F. Allen. 1997. Coding dialogues with the DAMSL annotation scheme. In *Working Notes: AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Cambridge.

Thomas H. Crystal and Arthur S. House. 1990. Articulation rate and the duration of syllables and stress groups in connected speech. *Journal of Acoustical Society of America*, 88:101–112.

Starkey Duncan. 1974. On the structure of speaker-auditor interaction during speaking turns. *Language in Society*, 2:161–180.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Curry I. Guinn. 1998. An analysis of initiative selection in collaborative task-oriented discourse. *User Modeling and User Adapted Interaction*, 8:255–314.

Susan Haller and Timothy Fossum. 1999. Using protocols to model mixed initiative interaction. In *Proceedings of AAAI Workshop on Mixed Initiative Intelligence*.

Peter A. Heeman and James F. Allen. 1995. The Trains spoken dialogue corpus. CD-ROM, Linguistics Data Consortium.

Peter A. Heeman, Fan Yang, Andrew L. Kun, and Alexander Shyrokov. 2005. Conventions in human-human multithreaded dialogues: A preliminary study. In *Proceedings of Intelligent User Interface (short paper session)*, pages 293–295, San Diego CA.

Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of CHI*, pages 159–166, Pittsburgh PA.

Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

Emanuel A. Schegloff. 2000. Overlapping talk and the organization of turn-taking for conversation. *Language in Society*, 29:1–63.

E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey. 2004. The ICSI meeting recorder dialog act corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*.

Kåre Sjölander and Jonas Beskow. 2000. WaveSurfer: An open source speech tool. In *Proceedings of ICSLP*, pages 4:464–467, Beijing China.

Ronnie W. Smith. 1993. Effective spoken natural language dialogue requires variable initiative behavior: an empirical study. In *AAAI93 Fall Symposium On Human-Computer Collaboration*.

Susan E. Strayer, Peter A. Heeman, and Fan Yang. 2003. Reconciling control and discourse structure. In J. Van Kuppevelt and R. W. Smith, editors, *Current and New Directions in Discourse and Dialogue*, chapter 14, pages 305–323. Kluwer Academic Publishers.

Siew Leng Toh, Fan Yang, and Peter A. Heeman. 2006. An annotation scheme for agreement analysis. In *Proceedings of INTERSPEECH*, Pittsburgh PA.

Marilyn Walker and Steve Whittaker. 1990. Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of 28th ACL*.

Steve Whittaker and Phil Stenton. 1988. Cues and control in expert-client dialogue. In *Proceedings of 28th ACL*, pages 123–130.

Fan Yang, Peter A. Heeman, Kristy Hollingshead, and Susan E. Strayer. 2007. Dialogueview: Annotating dialogues in multiple views with abstraction. *Natural Language Engineering*. To appear.

What Decisions Have You Made: Automatic Decision Detection in Conversational Speech

Pei-Yun Hsueh
School of Informatics
University of Edinburgh
Edinburgh EH9 8WL, UK
p.hsueh@ed.ac.uk

Johanna Moore
School of Informatics
University of Edinburgh
Edinburgh EH9 8WL, UK
J.Moore@ed.ac.uk

Abstract

This study addresses the problem of automatically detecting decisions in conversational speech. We formulate the problem as classifying decision-making units at two levels of granularity: dialogue acts and topic segments. We conduct an empirical analysis to determine the characteristic features of decision-making dialogue acts, and train MaxEnt models using these features for the classification tasks. We find that models that combine lexical, prosodic, contextual and topical features yield the best results on both tasks, achieving 72% and 86% precision, respectively. The study also provides a quantitative analysis of the relative importance of the feature types.

1 Introduction

Making decisions is an important aspect of conversations in collaborative work. In the context of meetings, the proposed argumentative models, e.g., in Pallotta et al. (2005) and Rienks et al. (2005), have specified decisions as an essential outcome of meetings. Whittaker et al. (2005) have also described how reviewing decisions is critical to the re-use of meeting recordings. For example, a new engineer who just get assigned to a project will need to know what major decisions have been made in previous meetings. Unless all decisions are recorded in meeting minutes or annotated in the speech recordings, it is difficult to locate the decision points by the browsing and playback utilities alone.

Banerjee and Rudnicky (2005) have shown that it is easier for users to retrieve the information they seek if the meeting record includes information about topic segmentation, speaker role, and meeting state (e.g., discussion, presentation, briefing). To assist users in identifying or revisiting decisions in meeting archives, our goal is to automatically identify the dialogue acts and segments where decisions are made. Because reviewing decisions is indispensable in collaborative work, automatic decision detection is expected to lend support to computer-assisted meeting tracking and understanding (e.g., assisting in the fulfilment of the decisions made in the meetings) and the development of group information management applications (e.g., constructing group memory).

2 Related Work

Spontaneous face-to-face dialogues in meetings violate many assumptions made by techniques previously developed for broadcast news (e.g., TDT and TRECVID), telephone conversations (e.g., Switchboard), and human-computer dialogues (e.g., DARPA Communicator). In order to develop techniques for understanding multiparty dialogues, smart meeting rooms have been built at several institutes to record large corpora of meetings in natural contexts, including CMU (Waibel et al., 2001), LDC (Cieri et al., 2002), NIST (Garofolo et al., 2004), ICSI (Janin et al., 2003), and in the context of the IM2/M4 project (Marchand-Mailet, 2003). More recently, scenario-based meetings, in which participants are assigned to different roles and given specific tasks, have been recorded in the context of the CALO project (the Y2 Scenario Data) (CALO, 2003) and the AMI project (Carletta et al., 2005).

The availability of meeting corpora has enabled researchers to begin to develop descriptive models of meeting discussions. Some researchers are modelling the dynamics of the meeting, exploiting dialogue models previously proposed for dialogue management. For example, Niekrasz et al. (2005) use the Issue-Based Information System (IBIS) model (Kunz and Ritte, 1970) to incorporate the history of dialogue moves into the Multi-Modal Discourse (MMD) ontology. Other researchers are modelling the content of the meeting using the type of structures proposed in work on argumentation. For example, Rienks et al. (2005) have developed an argument diagramming scheme to visualize the relations (e.g., positive, negative, uncertain) between utterances (e.g., statement, open issue), and Marchand et al. (2003) propose a schema to model different argumentation acts (e.g., accept, request, reject) and their organization and synchronization. Decisions are often seen as a by-product of these models.

Automatically extracting these argument models is a challenging task. However, researchers have begun to make progress towards this goal. For example, Gatica et al. (2005) and Wrede and Shriberg (2003) automatically identify the level of emotion in meeting spurts (e.g., group level of interest, hot spots). Other researchers have developed models for detecting agreement and disagreement in meetings, using models that combine lexical features with prosodic features (e.g., pause, duration, F0, speech rate) (Hillard et al., 2003) and structural information (e.g., the previous and following speaker) (Galley et al., 2004). More recently, Purver et al. (2006) have tackled the problem of detecting one type of decision, namely action items, which embody the transfer of group responsibility. However, no prior work has addressed the problem of automatically identifying decision-making units more generally in multiparty meetings. Moreover, no previous research has provided a quantitative account of the effects of different feature types on the task of automatic decision detection.

3 Research Goal

Our aim is to develop models for automatically detecting segments of conversation that contain decisions directly from the audio recordings and transcripts of the meetings, and to identify the feature combinations that are most effective for this task.

Meetings can be viewed at different levels of granularity. In this study, we first consider how to detect the dialogue acts that contain decision-related information (DM DAs). Since it is often difficult to interpret a decision without knowing the current topic of discussion, we are also interested in detecting decision-making segments at a coarser level of granularity: topic segments. The task of automatic decision detection can therefore be divided into two subtasks: detecting DM DAs and detecting decision-making topic segments (DM Segments).

In this study we propose to first empirically identify the features that are most characteristic of decision-making dialogue acts and then computationally integrate the characteristic features to locate the DM DAs in meeting archives. For the latter task, previous research on automatic meeting understanding and tracking has commonly utilized a classification framework, in which variants of generative and conditional models are computed directly from data. In this study, we use a Maximum Entropy (MaxEnt) classifier to combine the decision characteristic features to predict DM DAs and DM Segments.

4 Data

4.1 Decision Annotation

In this study, we use a set of 50 scenario-driven meetings (approximately 37,400 dialogue acts) that have been segmented into dialogue acts and annotated with decision information in the AMI meeting corpus. These meetings are driven by a scenario, wherein four participants play the role of Project Manager, Marketing Expert, Industrial Designer, and User Interface Designer in a design team in a series of four meetings. Each series of meeting recordings uses four distinctive speakers different from other series. The corpus includes manual transcripts for all meetings. It also comes with individual sound files recorded by close-talking head-mounted microphones and cross-talking sound files recorded by desktop microphones.

4.1.1 Decision-Making Dialogue Acts

In fact, it is difficult to determine whether a dialogue act contains information relevant to any decision point without knowing what decisions have been made in the meeting. Therefore, in this study DM DAs are annotated in a two-phase process: First, annotators are asked to browse through the meeting record and write an abstractive summary directed to the project manager about the decisions that have been made in the meeting. Next, another group of three annotators are asked to produce extractive summaries by selecting a subset (around 10%) of dialogue acts which form a summary of this meeting for the absent manager to understand what has transpired in the meeting.

Finally, this group of annotators are asked to go through the extractive dialogue acts one by one and judge whether they support any of the sentences in the decision section of the abstractive summary; if a dialogue act is related to any sentence in the decision section, a “decision link” from the dialogue act to the decision sentence is added. For those extracted dialogue acts that do not have any closely related sentence, the annotators are not obligated to specify a link. We then label the dialogue acts that have one or more decision links as DM DAs.

In the 50 meetings we used for the experiments, the annotators have on average found four decisions per meeting and specified around two decision links to each sentence in the decision summary section. Overall, 554 out of 37,400 dialogue acts have been annotated as DM DAs, accounting for 1.4% of all dialogue acts in the data set and 12.7% of the orginal extractive summary (which is consisted of the extracted dialogue acts). An earlier analysis has established the intercoder reliability of the two-phase process at the level of kappa ranging from 0.5 to 0.8. In this round of experiment, for each meeting in the 50-meeting dataset we randomly choose the DM DA annotation of one annotator as the sourec of its ground truth data.

4.1.2 Decision-Making Topic Segments

Topic segmentation has also been annotated for the AMI meeting corpus. Annotators had the freedom to mark a topic as subordinated (down to two levels) wherever appropriate. As the AMI meetings are scenario-driven, annotators are expected to find that most topics recur. Therefore, they are given a standard set of topic descriptions that can be used as labels for each identified topic segment. Annotators will only add a new label if they cannot find a match in the standard set. The AMI scenario meetings contain around 14 topic segments per meeting. Each segment lasts on average 44 dialogue acts long and contains two DM DAs.

DM Segments are operationalized as topic segments that contain one or more DM DAs. Overall, 198 out of 623 (31.78%) topic segments in the 50-meeting dataset are DM Segments. As the meetings we use are driven by a predetermined agenda, we expect to find that interlocutors are more likely to reach decisions when certain topics are brought up. Analysis shows that some topics are indeed more likely to contain decisions than others. For example, 80% of the segments labelled as Costing and 58% of those labelled Budget are DM Segments, whereas only 7% of the Existing Product segments and none of the Trend-Watching segments are DM Segments. Functional segments, such as Chitchat, Opening and Closing, almost never include decisions.

4.2 Features Used

To provide a qualitative account of the effect of different feature types on the task of automatic decision detection, we have conducted empirical analysis on four major types of features: lexical, prosodic, contextual and topical features.

4.2.1 Lexical Features

Previous research has studied lexical differences (i.e., occurrence counts of N-grams) between various aspects of speech, such as topics (Hsueh and Moore, 2006), speaker gender (Boulis and Ostendorf, 2005), and story-telling conversation (Gordon and Ganesan, 2005). As we expect that lexical differences also exist in DM conversations, we generated language models from the DM Dialogue Acts in the corpus. The comparison of the language models generated from the DM dialogue Acts and the rest of the conversations shows that some differences exist between the two models: (1) decision making conversations are more likely to contain *we* than *I* and *You*; (2) in decision-making conversations there are more explicit mentions of topical words, such as *advanced chips* and *functional design*; (3) in decision-

Type	Feature
Duration	Number of words spoken in current, previous and next subdialogue Duration (in seconds) of current, previous and next subdialogue
Pause	Amount of silence (in seconds) preceding a subdialogue Amount of silence (in seconds) following a subdialogue
Speech rate	Number of words spoken per second in current, previous and next subdialogue Number of syllables per second in current, previous and next subdialogue
Energy	Overall energy level Average energy level in the first, second, third, and fourth quarter of a subdialogue
Pitch	Maximum and minimum F0, overall slope and variance Slope and variance at the first 100 and 200 ms and last 100 and 200 ms, at the first and second half, and at each quarter of the subdialogue

Table 1: *Prosodic features used in this study.*

making conversations, there are fewer negative expressions, such as *I don't think* and *I don't know*. In an exploratory study using unigrams, as well as bigrams and trigrams, we found that using bigrams and trigrams does not improve the accuracy of classifying DM DAs, and therefore we include only unigrams in the set of lexical features in the experiments reported in Section 6.

4.2.2 Prosodic Features

Functionally, prosodic features, i.e., energy, and fundamental frequency (F0), are indicative of segmentation and saliency. In this study, we follow Shriberg and Stolcke's (2001) direct modelling approach to manifest prosodic features as duration, pause, speech rate, pitch contour, and energy level. We utilize the individual sound files provided in the AMI corpus. To extract prosodic features from the sound files, we use the Snack Sound Toolkit to compute a list of pitch and energy values delimited by frames of 10 ms, using the normalized cross correlation function. Then we apply a piecewise linearisation procedure to remove the outliers and average the linearised values of the units within the time frame of a word. Pitch contour of a dialogue act is approximated by measuring the pitch slope at multiple points within the dialogue act, e.g., the first and last 100 and 200 ms. The rate of speech is calculated as both the number of words spoken per second and the number of syllables per second. We use Festival's speech synthesis front-end to return phonemes and syllabification information. An exploratory study has shown the benefits of including immediate prosodic contexts, and thus we also include prosodic features of the immediately preceding and following dialogue acts. Table 1 contains a list of automatically generated prosodic features used in this study.

4.2.3 Contextual Features

From our qualitative analysis, we expect that contextual features specific to the AMI corpus, such as the speaker role (i.e., PM, ME, ID, UID) and meeting type (i.e., kick-off, conceptual design, functional design, detailed design) to be characteristic of the DM DAs. Analysis shows that (1) participants assigned to the role of PM produce 42.5% of the DM DAs, and (2) participants make relatively fewer decisions in the kick-off meetings. Analysis has also demonstrated a difference in the type, the reflexivity[1] and the number of addressees, between the DM DAs and the non-DM DAs. For example, dialogue acts of type ***inform***, ***suggest***, ***elicit assessment*** and ***elicit inform*** are more likely to be DM DAs.

We have also found that immediately preceding and following dialogue acts are important for identifying DM DAs. For example, ***stalls*** and ***fragments*** preceding and ***fragments*** following a DM DA are more likely than for non-DM DAs.[2] In

[1]According to the annotation guideline, the reflexivity reflects on how the group is carrying on the task. In this case, the interlocutors pause to evaluate the group performance less often when it comes to decision making.

[2]STALL is where people start talking before they are ready, or keep speaking when they haven't figured out what to say; FRAGMENT is the segment which is not really speech or is unclear enough to be transcribed, or where the speaker did not

contrast, there is a lower chance of seeing ***suggest*** and elicit-type DAs (i.e., ***elicit-inform***, ***elicit-suggestion***, ***elicit-assessment***) in the preceding and following DM DAs.

4.2.4 Topical Features

As reported in Section 4.1.2, we find that interlocutors are more likely to reach decisions when certain topics are brought up. Also, we expect decision-making conversations to take place towards the end of a topic segment. Therefore, in this study we include the following features: the label of the current topic segment, the position of the DA in a topic segment (measured in words, in seconds, and in %), the distance to the previous topic shift (both at the top-level and sub-topic level)(measured in seconds), the duration of the current topic segment (both at the top-level and sub-topic level)(measured in seconds).

5 Experiment

5.1 Classifying DM DAs

Detecting DM DAs is the first step of automatic decision detection. For this purpose, we trained MaxEnt models to classify each unseen sample as either DM DA (POS) or non-DM DA (NEG). We performed a 5-fold cross validation on the set of 50 meetings. In each fold, we trained MaxEnt models from the feature combinations in the training set, wherein each of the extracted dialogue acts has been labelled as either POS or NEG. Then, the models were used to classify unseen instances in the test set as either POS or NEG. In Section 4.2, we described the four major types of features used in this study: unigrams (LX1), prosodic (PROS), contextual (CONT), and topical (TOPIC) features. For comparison, we report the naive baseline obtained by training the models on the prosodic features alone, since the prosodic features can be generated fully automatically. The different combinations of features we used for training models can be divided into the following four groups: (A) using prosodic features alone (BASELINE), (B) using lexical, contextual and topical features alone (LX1, CONT, TOPIC); (C) using all available features except one of the four types of features (ALL-LX1, ALL-PROS, ALL-CONT, ALL-TOPIC); and

get far enough to express the intention.

(D) using all available features (ALL).

6 Results

6.1 Classifying DM Segments

Detecting DM segments is necessary for interpreting decisions, as it provides information about the current topic of discussion. Here we combine the predictions of the DM DAs to classify each unseen topic segment in the test set as either DM Segment (POS) or non-DM Segment (NEG). Recall that we defined a DM Segment as a segment that contains one or more hypothesized DM DAs. The task of detecting DM Segments can thus be viewed as that of detecting DM Dialogue Acts in a wider window.

6.2 EXP1: Classifying DM DAs

Table 2 reports the performance on the test set. The results show that models trained with all features (ALL), including lexical, prosodic, contextual and topical features, yield substantially better performance than the baseline on the task of detecting DM DAs. We carried out a one-way ANOVA to examine the effect of different feature combinations on overall accuracy (F1). The ANOVA suggests a reliable effect of feature type $(F(9,286) = 3.44; p < 0.001)$. Rows 2-4 in Table 2 report the performance of models in Group B that are trained with a single type of feature. Lexical features are the most predictive features when used alone. We performed sign tests to determine whether there are statistical differences among these models and the baseline. We find that when used alone, only lexical features (LX1) can train a better model than the baseline $(p < 0.001)$. However, none of these models yields a comparable performance to the ALL model.

To study the relative effect of the different feature types, Rows 5-8 in the table report the performance of models in Group C, which are trained with all available features except LX1, PROS, CONT and TOPIC features respectively. The amount of degradation in the overall accuracy (F1) of each of the models in relation to that of the ALL model indicates the contribution of the feature type that has been left out of the model. We performed sign tests to examine the differences among these models and the ALL model. We find that the ALL model outperforms all of these models $(p < 0.001)$ except

	Exact Match			Lenient Match		
Accuracy	Precision	Recall	F1	Precision	Recall	F1
BASELINE(PROS)	0.32	0.06	0.1	0.32	0.1	0.15
LX1	0.53	0.3	0.38	0.6	0.43	0.5
CONT	0	0	0	0	0	0
TOPIC	0.49	0.11	0.17	0.57	0.11	0.17
ALL-PROS	0.63	0.47	0.54	0.71	0.57	0.63
ALL-LX1	0.61	0.34	0.44	0.65	0.43	0.52
ALL-CONT	0.66	0.62	0.64	0.69	0.68	0.69
ALL-TOPIC	0.72	0.54	0.62	0.7	0.52	0.59
ALL	0.72	0.54	0.62	0.76	0.64	0.7

Table 2: *Effects of different combinations of features on detecting DM DAs.*

the model trained by leaving out contextual features (ALL-CONT). A closer investigation of the precision and recall of the ALL-CONT model shows that the contextual features are detrimental to recall but beneficial for precision. The mixed result is due to the fact that models trained with contextual features are tailored to recognize particular types of DM dialogue acts. Therefore, using these contextual features improves the precision for these types of DM DAs but reduces the overall recognition accuracy.

The last three columns of Table 2 are the results obtained using a lenient match measure, allowing a window of 10 seconds preceding and following a hypothesized DM DA for recognition. The better results show that there is room for ambiguity in the assessment of the exact timing of DM DAs.

6.3 EXP2: Classifying DM Segments

As expected, the results in Table 3 are better than those reported in Table 2, achieving at best 83% overall accuracy.The model that combines all features (ALL) yields significantly better results than the baseline. The ANOVA shows a reliable effect of different feature types on the task of detecting DM Segments $(F(11, 284) = 2.33; p <= 0.01)$. Rows 2-4 suggest that lexical features are the most predictive in terms of overall accuracy. Sign tests confirm the advantage of using lexical features (LX1) over the baseline (PROS) $(p < 0.05)$. Interestingly, the model that is trained with topical features alone (TOPIC) yields substantially better precision $(p < 0.001)$. The increase from 49% precision for the task of detecting DM DAs (in Table 2) to 91% for that of detecting DM Segments stems from the fact that decisions are more likely to occur in certain types of topic segments. In turn, training models with topical features helps eliminate incorrect predictions of DM DAs in these types of topic segments. However, the accuracy gain of the TOPIC model on detecting certain types of DM Segments does not extend to all types of DM Segments. This is shown by the significantly lower recall of the TOPIC model over the baseline $(p < 0.001)$.

Finally, Rows 5-8 report the performance of the models in Group (C) on the task of detecting DM Segments. Sign tests again show that the model that is trained with all available features (ALL) outperforms the models that leave out lexical, prosodic, or topical features $(p < 0.05)$. However, the ALL model does not outperform the model that leaves out contextual features. In addition, the contextual features degrade the recall but improve the precision on the task of detecting DM Segments. Calculating how much the overall accuracy of the models in Group C degrades from the ALL model shows that the most predictive features are the lexical features, followed by the topical and prosodic features.

7 Discussion

As suggested by the mixed results obtained by the model that is trained without the contextual features, the two-phase decision annotation procedure (as described in Section 4.1) may have caused annotators to select dialogue acts that serve different functional roles in a decision-making process in the set of DM DAs. For example, in the dialogue shown

	Exact Match		
Accuracy	Precision	Recall	F1
BASELINE(PROS)	0.67	0.39	0.49
LX1	0.69	0.69	0.69
CONT	0	0	0
TOPIC	0.91	0.17	0.29
ALL-PROS	0.82	0.76	0.79
ALL-LX1	0.79	0.64	0.7
ALL-CONT	0.79	0.86	0.83
ALL-TOPIC	0.75	0.73	0.74
ALL	0.86	0.8	0.82

Table 3: *Effects of different combinations of features on detecting DM Segments.*

(1) A: but um the feature that we considered for it not getting lost.
(2) B: Right. Well
(3) B: were talking about that a little bit
(4) B: when we got that email
(5) B: and we think that each of these are so distinctive, that it it's not just like another piece of technology around your house.
(6) B: It's gonna be somewhere that it can be seen.
(7) A: Mm-hmm.
(8) B: So we're we're not thinking that it's gonna be as critical to have the loss
(9) D: But if it's like under covers or like in a couch you still can't see it.
...
(10) A: Okay , that's a fair evaluation.
(11) A: Um we so we do we've decided not to worry about that for now.

Figure 1: Example decision-making discussion

in Figure 1, the annotators have marked dialogue act (1), (5), (8), and (11) as the DM DAs related to this decision: *"There will be no feature to help find the remote when it is misplaced"*. Among the four DM DAs, (1) describes the topic of what this decision is about; (5) and (8) describe the arguments that support the decision-making process; (11) indicates the level of agreement or disagreement for this decision. Yet these DM DAs which play different functional roles in the DM process may each have their own characteristic features. Training one model to recognize DM DAs of all functional roles may have degraded the performance on the classification tasks. Developing models for detecting DM DAs that play different functional roles requires a larger scale study to discover the anatomy of general decision-making discussions.

8 Conclusions and Future Work

This is the first study that aimed to detect segments of the conversation that contain decisions. We have (1) empirically analyzed the characteristic features of DM dialogue acts, and (2) computational developed models to detect DM dialogue acts and DM topic segments, given the set of characteristic features. Empirical analysis has provided a qualitative account of the DM-characteristic features, whereas training the computational models on different feature combinations has provided a quantitative account of the effect of different feature types on the task of automatic decision detection. Empirical analysis has exhibited demonstrable differences in the words (e.g., *we*), the contextual features (e.g., *meeting type*, *speaker role*, *dialogue act type*), and the topical features. The experimental results have suggested that (1) the model combining all the available features performs substantially better, achieving 62% and 82% overall accuracy on the task of detecting DM DAs and that of detecting DM Segments, respectively, (2) lexical features are the best indicators for both the task of detecting DM DAs and that of detecting DM Segments, and (3) combining topical features is important for improving the precision for the task of detecting DM Segments.

Many of the features used in this study require human intervention, such as manual transcriptions, annotated dialogue act segmentations and labels, annotated topic segmentations and labels, and other types of meeting-specific features. Our ultimate goal is to identify decisions using automatically induced features. Therefore, studying the performance degradation when using the automatically generated versions of these features (e.g., ASR words) is essential for developing a fully automated component on detecting decisions immediately after a meeting or even for when a meeting is still in progress. Another problem that has been pointed out in Section 6 and in Section 7 is the different functional roles of DM dialogue acts in current annotations. Purver et al. (2006) have suggested a hierarchical annotation scheme to accommodate the different aspects of action items. The same technique may be applicable in a more general decision detection task.

9 Acknowledgement

This work was supported by the European Union Integrated Project AMI (Augmented Multi-party Interaction, FP6-506811, publication AMI-204).

References

S. Banerjee, C. Rose, and A. I. Rudnicky. 2005. The necessity of a meeting recording and playback system, and the benefit of topic-level annotations to meeting browsing. In *Proceedings of the Tenth International Conference on Human-Computer Interaction*.

C. Boulis and M. Ostendorf. 2005. A quantitative analysis of lexical differences between genders in telephone conversation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*. ACM Press.

CALO. 2003. http://www.ai.sri.com/project/calo.

J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner. 2005. The ami meeting corpus: A pre-announcement. In *Proceedings of 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*.

C. Cieri, D. Miller, and K. Walker. 2002. Research methodologies, observations and outcomes in conversational speech data collection. In *Proceedings of the Human Language Technologies Conference (HLT)*.

M. Galley, J. McKeown, J. Hirschberg, and E. Shriberg. 2004. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Annual Meeting of the ACL*.

J. S. Garofolo, C. D. Laprun, M. Michel, V.M. Stanford, and E. Tabassi. 2004. The nist meeting room pilot corpus. In *Proceedings of LREC04*.

D. Gatica-Perez, I. McCowan, D. Zhang, and S. Bengio. 2005. Detecting group interest level in meetings. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*.

Andrew S. Gordon and Kavita Ganesan. 2005. Automated story extraction from conversational speech. In *Proceedings of the Third International Conference on Knowledge Capture (K-CAP 05)*.

D. Hillard, M. Ostendorf, and E. Shriberg. 2003. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proc. HLT-NAACL*.

P. Hsueh and J. Moore. 2006. Automatic topic segmentation and lablelling in multiparty dialogue. In *the first IEEE/ACM workshop on Spoken Language Technology (SLT)*. IEEE/ACM.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The icsi meeting corpus. In *Proceedings of ICASSP-2003*, Hong Kong.

W. Kunz and H. W. J. Ritte. 1970. Issue as elements of information system. Technical Report Working Paper 131, Institute of Urban and Regional Development Research, University of California, Berkeley.

S. Marchand-Mailet. 2003. Meeting record modeling for enhanced browsing. Technical report, Computer Vision and Multimedia Lab, Computer Centre, University of Geneva, Switzerland.

J. Niekrasz, M. Purver, J. Dowding, and S. Peters. 2005. Ontology-based discourse understanding for a persistent meeting assistant. In *Proc. of the AAAI Spring Symposium*.

V. Pallotta, J. Niekrasz, and M. Purver. 2005. Collaborative and argumentative models of meeting discussions. In *Proceeding of CMNA-05 workshop on Computational Models of Natural Arguments in IJCAI 05*.

M. Purver, P. Ehlen, and J. Niekrasz. 2006. Shallow discourse structure for action item detection. In *the Workshop of HLT-NAACL: Analyzing Conversations in Text and Speech*. ACM Press.

R. J. Rienks, D. Heylen, and E. van der Weijden. 2005. Argument diagramming of meeting conversations. In *Multimodal Multiparty Meeting Processing Workshop at the ICMI*.

E. Shriberg and A. Stolcke. 2001. Direct modeling of prosody: An overview of applications in automatic speech processing.

A. Waibel, M. Bett, F. Metze, K. Ries, T. Schaaf amd T. Schultz, H. Soltau, H. Yu, and K. Zechner. 2001. Advances in automatic meeting record creation and access. In *Proceedings of ICASSP*.

S. Whittaker, R. Laban, and S. Tucker. 2005. Analysing meeting records: An ethnographic study and technological implications. In *Proceedings of MLMI 2005*.

B. Wrede and E. Shriberg. 2003. Spotting hot spots in meetings: Human judgements and prosodic cues. In *Proceedings of EUROSPEECH 2003*.

Automatic Evaluation of Machine Translation Based on Rate of Accomplishment of Sub-goals

Kiyotaka Uchimoto and **Katsunori Kotani** and **Yujie Zhang** and **Hitoshi Isahara**
National Institute of Information and Communications Technology
3-5, Hikari-dai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
{uchimoto,yujie,isahara}@nict.go.jp, kat@khn.nict.go.jp

Abstract

The quality of a sentence translated by a machine translation (MT) system is difficult to evaluate. We propose a method for automatically evaluating the quality of each translation. In general, when translating a given sentence, one or more conditions should be satisfied to maintain a high translation quality. In English-Japanese translation, for example, prepositions and infinitives must be appropriately translated. We show several procedures that enable evaluating the quality of a translated sentence more appropriately than using conventional methods. The first procedure is constructing a test set where the conditions are assigned to each test-set sentence in the form of yes/no questions. The second procedure is developing a system that determines an answer to each question. The third procedure is combining a measure based on the questions and conventional measures. We also present a method for automatically generating sub-goals in the form of yes/no questions and estimating the rate of accomplishment of the sub-goals. Promising results are shown.

1 Introduction

In machine translation (MT) research, appropriately evaluating the quality of MT results is an important issue. In recent years, many researchers have tried to automatically evaluate the quality of MT and improve the performance of automatic MT evaluations (Niessen et al., 2000; Akiba et al., 2001; Papineni et al., 2002; NIST, 2002; Leusch et al., 2003; Turian et al., 2003; Babych and Hartley, 2004; Lin and Och, 2004; Banerjee and Lavie, 2005; Gimeńez et al., 2005) because improving the performance of automatic MT evaluation is expected to enable us to use and improve MT systems efficiently. For example, Och reported that the quality of MT results was improved by using automatic MT evaluation measures for the parameter tuning of an MT system (Och, 2003). This report shows that the quality of MT results improves as the performance of automatic MT evaluation improves.

MT systems can be ranked if a set of MT results for each system and their reference translations are given. Usually, about 300 or more sentences are used to automatically rank MT systems (Koehn, 2004). However, the quality of a sentence translated by an MT system is difficult to evaluate. For example, the results of five MTs into Japanese of the sentence “The percentage of stomach cancer among the workers appears to be the highest for any asbestos workers.” are shown in Table 1. A conventional automatic evaluation method ranks the fifth MT result first although its human subjective evaluation is the lowest. This is because conventional methods are based on the similarity between a translated sentence and its reference translation, and they give the translated sentence a high score when the two sentences are globally similar to each other in terms of lexical overlap. However, in the case of the above example,

Table 1: Examples of conventional automatic evaluations.

Original sentence	The percentage of stomach cancer among the workers appears to be the highest for any asbestos workers.
Reference translation (in Japanese)	*roudousha no igan no wariai wa , asubesuto roudousha no tame ni saikou to naru youda .*

System	MT results	BLEU	NIST	Fluency	Adequacy
1	*roudousha no aida no igan no paasenteeji wa , donoyouna ishiwata roudousha no tame ni demo mottomo ookii youdearu .*	0.2111	2.1328	2	3
2	*roudousha no aida no igan no paasenteeji wa, arayuru asubesuto roudousha no tame ni mottomo takai youni omowa re masu .*	0.2572	2.1234	2	3
3	*roudousha no aida no igan no paasenteeji wa donna asubesuto no tame ni mo mottomo takai youni mie masu*	0	1.8094	1	2
4	*roudousha no aida no igan no paasenteeji wa nin'ino ishiwata ni wa mottomo takaku mie masu .*	0	1.5902	1	2
5	*roudousha no naka no igan no wariai wa donna asubesuto ni mo mottomo takai youni mieru .*	0.2692	2.2640	1	2

the most important thing to maintain a high translation quality is to correctly translate "for" into the target language, and it would be difficult to detect the importance just by comparing an MT result and its reference translations even if the number of reference translations is increased.

In general, when translating a given sentence, one or more conditions should be satisfied to maintain a high translation quality. In this paper, we show that constructing a test set where the conditions that are mainly established from a linguistic point of view are assigned to each test-set sentence in the form of yes/no questions, developing a system that determines an answer to each question, and combining a measure based on the questions and conventional measures enable the evaluation of the quality of a translated sentence more appropriately than using conventional methods. We also present a method for automatically generating sub-goals in the form of yes/no questions and estimating the rate of accomplishment of the sub-goals.

2 Test Set for Evaluating Machine Translation Quality

2.1 Test Set

Two main types of data are used for evaluating MT quality. One type of data is constructed by arbitrarily collecting sentence pairs in the source- and target-languages, and the other is constructed by intensively collecting sentence pairs that include linguistic phenomena that are difficult to automatically translate. Recently, MT evaluation campaigns such as the International Workshop on Spoken Language Translation [1], NIST Machine Translation Evaluation [2], and HTRDP Evaluation [3] were organized to support the improvement of MT techniques. The data used in the evaluation campaigns were arbitrarily collected from newspaper articles or travel conversation data for fair evaluation. They are classified as the former type of data mentioned above. On the other hand, the data provided by NTT (Ikehara et al., 1994) and that constructed by JEIDA (Isahara, 1995) are classified as the latter type. Almost all the data mentioned above consist of only parallel translations in two languages. Data with information for evaluating MT results, such as JEIDA's are rarely found. In this paper, we call data that consist of parallel translations collected for MT evaluation and that the information for MT evaluation is assigned to, a *test set*.

The most characteristic information assigned to the JEIDA test set is the yes/no question for assessing the translation results. For example, a yes/no question such as "Is 'for' translated into an expression representing a cause/reason such as 'de'?" (in Japanese) is assigned to a test-set sentence. We can evaluate MT results objectively by answering the question. An example of a test-set sample consisting of an ID, a source-language sample sentence, its reference translation, and a question is as follows.

[1]http://www.slt.atr.jp/IWSLT2006/
[2]http://www.nist.gov/speech/tests/mt/index.htm
[3]http://www.863data.org.cn/

ID	1.1.7.1.3-1
Sample sentence	The percentage of stomach cancer among the workers appears to be the highest for any asbestos workers.
Reference translation (in Japanese)	*roudousha no igan no wariai wa , asubesuto roudousha no tame ni saikou to naru youda .*
Question	Is "appear to" translated into an auxiliary verb such as "*youda*"?

The questions are classified mainly in terms of grammar, and the numbers to the left of the hyphenation of each ID such as 1.1.7.1.3 represent the categories of the questions. For example, the above question is related to catenative verbs.

The JEIDA test set consists of two parts, one for the evaluation of English-Japanese MT and the other for that of Japanese-English MT. We focused on the part for English-Japanese MT. This part consists of 769 sample sentences, each of which has a yes/no question.

The 769 sentences were translated by using five commercial MT systems to investigate the relationship between subjective evaluation based on yes/no questions and conventional subjective evaluation based on fluency and adequacy. The instruction for the subjective evaluation based on fluency and adequacy followed that given in the TIDES specification (TIDES, 2002). The subjective evaluation based on yes/no questions was done by manually answering each question for each translation. The subjective evaluation based on the yes/no questions was stable; namely, it was almost independent of the human subjects in our preliminary investigation. There were only two questions for which the answers generated inconsistency in the subjective evaluation when 1,500 question-answer pairs were randomly sampled and evaluated by two human subjects.

Then, we investigated the correlation between the two types of subjective evaluation. The correlation coefficients mentioned in this paper are statistically significant at the 1% or less significance level. The Spearman rank-order correlation coefficient is used in this paper. In the subjective evaluation based on yes/no questions, yes and no were numerically transformed into 1 and -1. For 3,845 translations obtained by using five MT systems, the correlation coefficients between the subjective evaluations based on yes/no questions and based on fluency and adequacy were 0.48 for fluency and 0.63 for adequacy. These results indicate that the two subjective evaluations have relatively strong correlations. The correlation is especially strong between the subjective evaluation based on yes/no questions and adequacy.

2.2 Expansion of JEIDA Test Set

Each sample sentence in the JEIDA test set has only one question. Therefore, in the subjective evaluation using the JEIDA test set, translation errors that do not involve the pre-assigned question are ignored even if they are serious. Therefore, translations that have serious errors that are not related to the question tend to be evaluated as being of high quality. To solve this problem, we expanded the test set by adding new questions about translations with the serious errors.

Sentences whose average grades were three or less for fluency and adequacy for the translation results of the five MT systems were selected for the expansion. Besides them, sentences whose average grades were more than three for fluency and adequacy for the translation results of the five MT systems were selected when a majority of evaluation results based on yes/no questions about the translations of the five MT systems were no. The number of selected sentences was 150. The expansion was manually performed using the following steps.

1. Serious translation errors are extracted from the MT results.

2. For each extracted error, questions strongly related to the error are searched for in the test set. If related questions are found, the same types of questions are generated for the selected sentence, and the same ID as that of the related question is assigned to each generated question. Otherwise, questions are newly generated, and a new ID is assigned to each generated question.

3. Each MT result is evaluated according to each added question.

Eventually, one or more questions were assigned to each selected sentence in the test set. Among the 150

Table 2: Expanded test-set samples.

Original	ID	1.1.7.1.3-1
	Sample sentence	The percentage of stomach cancer among the workers appears to be the highest for any asbestos workers.
	Reference translation (in Japanese)	*roudousha no igan no wariai wa , asubesuto roudousha no tame ni saikou to naru youda* .
	Question (Q-0)	Is "appear to" translated into an auxiliary verb such as "*youda*"?
Expanded	ID	1.1.6.1.3-5
	Translation error	"For" is not translated appropriately.
	Question-1 (Q-1)	Is "for" translated into an expression representing a cause/reason such as "*. . . de*"?
Expanded	ID	Additional-1
	Translation error	Some expressions are not translated.
	Question-2 (Q-2)	Are all English words translated into Japanese?

Table 3: Examples of subjective evaluations based on yes/no questions.

System	MT results	Answer Q-0	Q-1	Q-2	Fluency	Adequacy
1	*roudousha no aida no igan no paasenteeji wa , donoyouna ishiwata roudousha no tame ni demo mottomo ookii youdearu .*	Yes	No	Yes	2	3
2	*roudousha no aida no igan no paasenteeji wa, arayuru asubesuto roudousha no tame ni mottomo takai youni omowa re masu .*	Yes	Yes	Yes	2	3
3	*roudousha no aida no igan no paasenteeji wa donna asubesuto no tame ni mo mottomo takai youni mie masu*	Yes	No	No	1	2
4	*roudousha no aida no igan no paasenteeji wa nin'ino ishiwata ni wa mottomo takaku mie masu .*	Yes	No	No	1	2
5	*roudousha no naka no igan no wariai wa donna asubesuto ni mo mottomo takai youni mieru .*	Yes	No	No	1	2

selected sentences, questions were newly assigned to 103 sentences. The number of added questions was 148. The maximum number of questions added to a sentence was five. After expanding the test set, the correlation coefficients between the subjective evaluations based on yes/no questions and based on fluency and adequacy increased from 0.48 to 0.51 for fluency and from 0.63 to 0.66 for adequacy. The differences between the correlation coefficients obtained before and after the expansion are statistically significant at the 5% or less significance level for adequacy. These results indicate that the expansion of the test set significantly improves the correlation between the subjective evaluations based on yes/no questions and based on adequacy. When two or more questions were assigned to a test-set sentence, the subjective evaluation based on the questions was decided by the majority answer. The majority answers, yes and no, were numerically transformed into 1 and -1. Ties between yes and no were transformed into 0. Examples of added questions and the subjective evaluations based on the questions are shown in Tables 2 and 3.

3 Automatic Evaluation of Machine Translation Based on Rate of Accomplishment of Sub-goals

3.1 A New Measure for Evaluating Machine Translation Quality

The JEIDA test set was not designed for automatic evaluation but for human subjective evaluation. However, a measure for automatic MT evaluation that strongly correlates fluency and adequacy is likely to be established because the subjective evaluation based on yes/no questions has a relatively strong correlation with the subjective evaluation based on fluency and adequacy, as mentioned in Section 2. In this section, we describe a method for automatically evaluating MT quality by predicting an answer to each yes/no question and using those answers.

Hereafter, we assume that each yes/no question is defined as a sub-goal that a given translation should satisfy and that the sub-goal is accomplished if the answer to the corresponding yes/no question to the sub-goal is yes. We also assume that the sub-goal is unaccomplished if the answer is no. A new evaluation score, A, is defined based on a multiple lin-

Table 4: Examples of Patterns.

Sample sentence	She lived there by herself.
Question	Is "by herself" translated as "*hitori de*"?
Pattern	The answer is *yes* if the pattern [*hitori dake de*|*hitori kiri de* |*tandoku de*|*tanshin de*] is included in a translation. Otherwise, the answer is *no*.
Sample sentence	They speak English in New Zealand.
Question	The personal pronoun "they" is omitted in a translation like "*nyuujiilando de wa eigo wo hanasu*"?
Pattern	The answer is *yes* if the pattern [*karera wa*|*sore ra wa*] is not included in a translation. Otherwise, the answer is *no*.

ear regression model as follows using the rate of accomplishment of the sub-goals and the similarities between a given translation and its reference translation. The best-fitted line for the observed data is calculated by the method of least-squares (Draper and Smith, 1981).

$$A = \sum_{i=1}^{m} \lambda_{S_i} \times S_i + \sum_{j=1}^{n} (\lambda_{Q_j} \times Q_j + \lambda_{Q_j^{'}} \times Q_j^{'}) + \lambda_{\epsilon} \quad (1)$$

$$Q_j = \begin{cases} 1 : \text{if subgoal is accomplished} \\ 0 : \text{otherwise} \end{cases} \quad (2)$$

$$Q_j^{'} = \begin{cases} 1 : \text{if subgoal is unaccomplished} \\ 0 : \text{otherwise} \end{cases} \quad (3)$$

Here, the term Q_j corresponds to the rate of accomplishment of the sub-goal having the i-th ID, and λ_{Q_j} is a weight for the rate of accomplishment. The term $Q_j^{'}$ corresponds to the rate of unaccomplishment of the sub-goal having the i-th ID, and $\lambda_{Q_j^{'}}$ is a weight for the rate of unaccomplishment. The value n indicates the number of types of sub-goals. The term λ_ϵ is constant.

The term S_i indicates a similarity between a translated sentence and its reference translation, and λ_{S_i} is a weight for the similarity. Many methods for calculating the similarity have been proposed (Niessen et al., 2000; Akiba et al., 2001; Papineni et al., 2002; NIST, 2002; Leusch et al., 2003; Turian et al., 2003; Babych and Hartley, 2004; Lin and Och, 2004; Banerjee and Lavie, 2005; Gimeńez et al., 2005). In our research, 23 scores, namely BLEU (Papineni et al., 2002) with maximum n-gram lengths of 1, 2, 3, and 4, NIST (NIST, 2002) with maximum n-gram lengths of 1, 2, 3, 4, and 5, GTM (Turian et al., 2003) with exponents of 1.0, 2.0, and 3.0, METEOR (exact) (Banerjee and Lavie, 2005), WER (Niessen et al., 2000), PER (Leusch et al., 2003), and ROUGE (Lin, 2004) with n-gram lengths of 1, 2, 3, and 4 and 4 variants (LCS, S∗, SU∗, W-1.2), were used to calculate each similarity S_i. Therefore, the value of m in Eq. (1) was 23. Japanese word segmentation was performed by using JUMAN [4] in our experiments.

As you can see, the definition of our new measure is based on a combination of an evaluation measure focusing on local information and that focusing on global information.

3.2 Automatic Estimation of Rate of Accomplishment of Sub-goals

The rate of accomplishment of sub-goals is estimated by determining the answer to each question as yes or no. This section describes a method based on simple patterns for determining the answers.

An answer to each question is automatically determined by checking whether patterns are included in a translation or not. The patterns are constructed for each question. All of the patterns are expressed in *hiragana* characters. Before applying the patterns to a given translation, the translation is transformed into *hiragana* characters, and all punctuation is eliminated. The transformation to *hiragana* characters was performed by using JUMAN in our experiments.

Test-set sentences, the questions assigned to them, and the patterns constructed for the questions are shown in Table 4. In the patterns, the symbol "|" represents "OR".

3.3 Automatic Sub-goal Generation and Automatic Estimation of Rate of Accomplishment of Sub-goals

We found that expressions important for maintaining a high translation quality were often commonly

[4]http://www.kc.t.u-tokyo.ac.jp/nl-resource/juman.html

included in the reference translations for each test-set sentence. We also found that the expression was also related to the yes/no question assigned to the test-set sentence. Therefore, we automatically generate yes/no questions in the following steps.

1. For each test-set sentence, a set of words commonly appearing in the reference translations are extracted.

2. For each combination of n words in the set of words extracted in the first step, skip word n-grams commonly appearing in the reference translations in the same word order are selected as a set of common skip word n-grams.

3. For each test-set sentence, the sub-goal is defined as the yes/no question "Are all of the common skip word n-grams included in the translation?"

If no common skip word n-grams are found, the yes/no question is not generated. The answer to the yes/no question is determined to be yes if all of the common skip word n-grams are included in a translation. Otherwise, the answer is determined to be no.

This scheme assigns greater weight to important phrases that should be included in the translation to maintain a high translation quality. Our observation is that those important phrases are often common between human translations. A similar scheme was proposed by Babych and Hartley (Babych and Hartley, 2004) for BLEU. In their scheme, greater weight is assigned to components that are salient throughout the document. Therefore, their scheme focuses on global context while our scheme focuses on local context. We believe that the two schemes are complementary to each other.

4 Experiments and Discussion

In our experiments, the translation results of three MT systems and their subjective evaluation results were used as a development set for constructing the patterns described in Section 3.2 and for tuning the parameters λ_{S_i}, λ_{Q_j}, $\lambda_{Q'_j}$, and λ_ϵ in Eq. (1). The translations and evaluation results of the remaining two MT systems were used as an evaluation set for testing.

In the development set, each test-set sentence has at least one question, at least one reference translation, three MT results, and subjective evaluation results of the three MT results. The patterns for determining yes/no answers were manually constructed for the questions assigned to the 769 test-set sentences. There were 917 questions assigned to them. Among them, the patterns could be constructed for 898 questions assigned to 767 test-set sentences. The remaining 19 questions were skipped because making simple patterns as described in Section 3.2 was difficult; for example, one of the questions was "Is the whole sentence translated into one sentence?". The yes/no answer determination accuracies obtained by using the patterns are shown in Table 5.

Table 5: Results of yes/no answer determination.

Test set	Accuracy
Development	97.6% (2,629/2,694)
Evaluation	82.8% (1,487/1,796)

We investigated the correlation between the evaluation score, A in Eq. (1) and the subjective evaluations, fluency and adequacy, for the 769 test-set sentences. First, to maximize the correlation coefficients between the evaluation score, A, and the human subjective evaluations, fluency and adequacy, the optimal values of λ_{S_i}, λ_{Q_j}, $\lambda_{Q'_j}$, and λ_ϵ in Eq. (1) were investigated using the development set within a framework of multiple linear regression modeling (Draper and Smith, 1981). Then, the correlation coefficients were investigated by using the optimal value set. The results are shown in Table 6, 7, and 8. In these tables, "Conventional method" indicates the correlation coefficients obtained when A was calculated by using only similarities S_i. "Conventional method (combination)" is a combination of existing automatic evaluation methods from the literature. "Our method (automatic)" indicates the correlation coefficients obtained when the results of the automatic determination of yes/no answers were used to calculate Q_j and Q'_j in Eq. (1). For the 19 questions for which the patterns could not be constructed, Q_j was set at 0. "Our method (full automatic)" indicates the correlation coefficients obtained when the results of the automatic sub-goal generation and determination of rate of accomplish-

Table 6: Coefficients of correlation between evaluation score A and fluency/adequacy. (A reference translation is used to calculate S_i.)

Method	fluency		adequacy	
	Development set	Evaluation set	Development set	Evaluation set
Conventional method (WER)	0.43	0.48	0.42	0.48
Conventional method (combination)	0.52	0.51	0.49	0.47
Our method (automatic)	0.90$*$	0.59$*$	0.89$*$	0.62$*$
Our method (upper bound)	0.90$*$	0.62$*$	0.90$*$	0.68$*$

Table 7: Coefficients of correlation between evaluation score A and fluency/adequacy. (Three reference translations are used to calculate S_i.)

Method	fluency		adequacy	
	Development set	Evaluation set	Development set	Evaluation set
Conventional method (WER)	0.47	0.51	0.45	0.51
Conventional method (combination)	0.54	0.54	0.51	0.52
Our method (automatic)	0.90$*$	0.60$*$	0.90$*$	0.64$*$
Our method (full automatic)	0.85$*$	0.58	0.84$*$	0.60$*$
Our method (upper bound)	0.90$*$	0.62$*$	0.90$*$	0.69$*$

Table 8: Coefficients of correlation between evaluation score A and fluency/adequacy. (Five reference translations are used to calculate S_i.)

Method	fluency		adequacy	
	Development set	Evaluation set	Development set	Evaluation set
Conventional method (WER)	0.49	0.53	0.46	0.53
Conventional method (combination)	0.56	0.56	0.52	0.54
Our method (automatic)	0.90$*$	0.60	0.90$*$	0.63$*$
Our method (full automatic)	0.86$*$	0.59	0.85$*$	0.60$*$
Our method (upper bound)	0.91$*$	0.63$*$	0.90$*$	0.69$*$

In these tables, $*$ indicates significance at the 5% or less significance level.

ment of sub-goals were used to calculate Q_j and $Q_j^{'}$ in Eq. (1). Skip word trigrams, skip word bigrams, and skip word unigrams were used for generating the sub-goals according to our preliminary experiments. "Our method (upper bound)" indicates the correlation coefficients obtained when human judgments on the questions were used to calculate Q_j and $Q_j^{'}$.

As shown in Table 6, 7, and 8, our methods significantly outperform the conventional methods from literature. Note that WER outperformed other individual measures like BLEU and NIST in our experiments, and the combination of existing automatic evaluation methods from the literature outperformed individual lexical similarity measures by themselves in almost all cases. The differences between the correlation coefficients obtained using our method and the conventional methods are statistically significant at the 5% or less significance level for fluency and adequacy, even if the number of reference translations increases, except in three cases shown in Table 7 and 8. This indicates that considering the rate of accomplishment of sub-goals to automatically evaluate the quality of each translation is useful, especially when the number of reference translations is small.

The differences between the correlation coefficients obtained using two automatic methods are not significant. These results indicate that we can reduce the development cost for constructing sub-goals. However, there are still significant gaps between the correlation coefficients obtained using a fully automatic method and upper bounds. These gaps indicate that we need further improvement in automatic sub-goal generation and automatic estimation of rate of accomplishment of sub-goals, which is our future work.

Human judgments of adequacy and fluency are known to be noisy, with varying levels of intercoder agreement. Recent work has tended to apply cross-judge normalization to address this issue (Blatz et al., 2003). We would like to evaluate against the normalized data in the future.

5 Conclusion and Future Work

We demonstrated that the quality of a translated sentence can be evaluated more appropriately than by using conventional methods. That was demonstrated by constructing a test set where the conditions that should be satisfied to maintain a high translation quality are assigned to each test-set sentence in the form of a question, by developing a system that determines an answer to each question, and by combining a measure based on the questions and conventional measures. We also presented a method for automatically generating sub-goals in the form of yes/no questions and estimating the rate of accomplishment of the sub-goals. Promising results were obtained.

In the near future, we would like to expand the test set to improve the upper bound obtained by our method. We are also planning to expand the method and improve the accuracy of the automatic sub-goal generation and determination of the rate of accomplishment of sub-goals. The sub-goals of a given sentence should be generated by considering the complexity of the sentence and the alignment information between the original source-language sentence and its translation. Further advanced generation and estimation would give us information about the erroneous parts of MT results and their quality. We believe that future research would allow us to develop high-quality MT systems by tuning the system parameters based on the automatic MT evaluation measures.

Acknowledgments

The guideline for expanding the test set is based on that constructed by the Technical Research Committee of the AAMT (Asia-Pacific Association for Machine Translation) The authors would like to thank the committee members, especially, Mr. Kentaro Ogura, Ms. Miwako Shimazu, Mr. Tatsuya Sukehiro, Mr. Masaru Fuji, and Ms. Yoshiko Matsukawa for their cooperation. This research is partially supported by special coordination funds for promoting science and technology.

References

Yasuhiro Akiba, Kenji Imamura, and Eiichiro Sumita. 2001. Using Multiple Edit Distances to Automatically Rank Machine Translation Output. In *Proceedings of the MT Summit VIII*, pages 15–20.

Bogdan Babych and Anthony Hartley. 2004. Extending the BLEU MT Evaluation Method with Frequency Weightings. In *Proceedings of the 42nd ACL*, pages 622–629.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 65–72.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2003. Confidence Estimation for Machine Translation. Technical report, Center for Language and Speech Processing, Johns Hopkins University. Summer Workshop Final Report.

Norman R. Draper and Harry Smith. 1981. *Applied Regression Analysis. 2nd edition*. Wiley.

Jesuś Gimeńez, Enrique Amigo ́, and Chiori Hori. 2005. Machine translation evaluation inside qarla. In *Proceedings of the IWSLT'05*.

Satoru Ikehara, Satoshi Shirai, and Kentaro Ogura. 1994. Criteria for Evaluating the Linguistic Quality of Japanese to English Machine Translations. *Transactions of the JSAI*, 9(4):569–579. (in Japanese).

Hitoshi Isahara. 1995. JEIDA's Test-Sets for Quality Evaluation of MT Systems – Technical Evaluation from the Developer's Point of View.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on EMNLP*, pages 388–395.

Gregor Leusch, Nicola Ueffing, and Hermann Ney. 2003. A Novel String-to-String Distance Measure with Applications to Machine Translation Evaluation. In *Proceedings of the MT Summit IX*, pages 240–247.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of the 20th COLING*, pages 501–507.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*, pages 74–81.

Sonja Niessen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *Proceedings of the LREC 2000*, pages 39–45.

NIST. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. Technical report, NIST.

Franz Josef Och. 2003. Minimum Error Training in Statistical Machine Translation. In *Proceedings of the 41st 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 ACL*, pages 311–318.

TIDES. 2002. Linguistic Data Annotation Specification: Assessment of Fluency and Adequacy in Arabic-English and Chinese-English Translations. http://www.ldc.upenn.edu/Projects/TIDES/Translation/TransAssess02.pdf.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. 2003. Evaluation of Machine Translation and its Evaluation. In *Proceedings of the MT Summit IX*, pages 386–393.

Source-Language Features and Maximum Correlation Training for Machine Translation Evaluation

Ding Liu and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We propose three new features for MT evaluation: source-sentence constrained n-gram precision, source-sentence reordering metrics, and discriminative unigram precision, as well as a method of learning linear feature weights to directly maximize correlation with human judgments. By aligning both the hypothesis and the reference with the source-language sentence, we achieve better correlation with human judgments than previously proposed metrics. We further improve performance by combining individual evaluation metrics using maximum correlation training, which is shown to be better than the classification-based framework.

1 Introduction

Evaluation has long been a stumbling block in the development of machine translation systems, due to the simple fact that there are many correct translations for a given sentence. The most commonly used metric, BLEU, correlates well over large test sets with human judgments (Papineni et al., 2002), but does not perform as well on sentence-level evaluation (Blatz et al., 2003). Later approaches to improve sentence-level evaluation performance can be summarized as falling into four types:

- Metrics based on common loose sequences of MT outputs and references (Lin and Och, 2004; Liu and Gildea, 2006). Such metrics were shown to have better fluency evaluation performance than metrics based on n-grams such BLEU and NIST (Doddington, 2002).
- Metrics based on syntactic similarities such as the head-word chain metric (HWCM) (Liu and Gildea, 2005). Such metrics try to improve fluency evaluation performance for MT, but they heavily depend on automatic parsers, which are designed for well-formed sentences and cannot generate robust parse trees for MT outputs.
- Metrics based on word alignment between MT outputs and the references (Banerjee and Lavie, 2005). Such metrics do well in adequacy evaluation, but are not as good in fluency evaluation, because of their unigram basis (Liu and Gildea, 2006).
- Combination of metrics based on machine learning. Kulesza and Shieber (2004) used SVMs to combine several metrics. Their method is based on the assumption that higher classification accuracy in discriminating human- from machine-generated translations will yield closer correlation with human judgment. This assumption may not always hold, particularly when classification is difficult. Lita et al. (2005) proposed a log-linear model to combine features, but they only did preliminary experiments based on 2 features.

Following the track of previous work, to improve evaluation performance, one could either propose new metrics, or find more effective ways to combine the metrics. We explore both approaches. Much work has been done on computing MT scores based

Proceedings of NAACL HLT 2007, pages 41–48,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

on the pair of MT output/reference, and we aim to investigate whether some other information could be used in the MT evaluation, such as source sentences. We propose two types of source-sentence related features as well as a feature based on part of speech. The three new types of feature can be summarized as follows:

- Source-sentence constrained n-gram precision. Overlapping n-grams between an MT hypothesis and its references do not necessarily indicate correct translation segments, since they could correspond to different parts of the source sentence. Thus our constrained n-gram precision counts only overlapping n-grams in MT hypothesis and reference which are aligned to the same words in the source sentences.

- Source-sentence reordering agreement. With the alignment information, we can compare the reorderings of the source sentence in the MT hypothesis and in its references. Such comparison only considers the aligned positions of the source words in MT hypothesis and references, and thus is oriented towards evaluating the sentence structure.

- Discriminative unigram precision. We divide the normal n-gram precision into many sub-precisions according to their part of speech (POS). The division gives us flexibility to train the weights of each sub-precision in frameworks such as SVM and Maximum Correlation Training, which will be introduced later. The motivation behind such differentiation is that different sub-precisions should have different importance in MT evaluation, e.g., sub-precision of nouns, verbs, and adjectives should be important for evaluating adequacy, and sub-precision in determiners and conjunctions should mean more in evaluating fluency.

Along the direction of feature combination, since indirect weight training using SVMs, based on reducing classification error, cannot always yield good performance, we train the weights by directly optimizing the evaluation performance, i.e., maximizing the correlation with the human judgment. This type of direct optimization is known as Minimum Error Rate Training (Och, 2003) in the MT community, and is an essential component in building the state-of-art MT systems. It would seem logical to apply similar methods to MT evaluation. What is more, Maximum Correlation Training (MCT) enables us to train the weights based on human fluency judgments and adequacy judgments respectively, and thus makes it possible to make a fluency-oriented or adequacy-oriented metric. It surpasses previous MT metrics' approach, where a a single metric evaluates both fluency and adequacy. The rest of the paper is organized as follows: Section 2 gives a brief recap of n-gram precision-based metrics and introduces our three extensions to them; Section 3 introduces MCT for MT evaluation; Section 4 describes the experimental results, and Section 5 gives our conclusion.

2 Three New Features for MT Evaluation

Since our source-sentence constrained n-gram precision and discriminative unigram precision are both derived from the normal n-gram precision, it is worth describing the original n-gram precision metric, BLEU (Papineni et al., 2002). For every MT hypothesis, BLEU computes the fraction of n-grams which also appear in the reference sentences, as well as a brevity penalty. The formula for computing BLEU is shown below:

$$\text{BLEU} = \frac{\text{BP}}{N} \sum_{n=1}^{N} \frac{\sum_C \sum_{ngram \in C} Count_{clip}(ngram)}{\sum_C \sum_{ngram' \in C} Count(ngram')}$$

where C denotes the set of MT hypotheses. $Count_{clip}(ngram)$ denotes the clipped number of n-grams in the candidates which also appear in the references. BP in the above formula denotes the brevity penalty, which is set to 1 if the accumulated length of the MT outputs is longer than the arithmetic mean of the accumulated length of the references, and otherwise is set to the ratio of the two. For sentence-level evaluation with BLEU, we compute the score based on each pair of MT hypothesis/reference. Later approaches, as described in Section 1, use different ways to manipulate the morphological similarity between the MT hypothesis and its references. Most of them, except NIST, consider the words in MT hypothesis as the same, i.e., as long as the words in MT hypothesis appear in the references, they make no difference to the metrics.[1] NIST computes the n-grams weights as the logarithm of the ratio of the n-gram frequency and its one word lower n-gram frequency. From our experiments, NIST is not generally better than BLEU, and the reason, we conjecture, is that it differentiates the n-grams too much and the frequency estimated upon the evaluation corpus is not always reliable. In this section we will describe two other strategies for differentiating the n-grams, one of which uses the alignments with the source sentence as a further constraint, while the other differentiates the n-gram precisions according to POS.

2.1 Source-sentence Constrained N-gram Precision

The quality of an MT sentence should be independent of the source sentence given the reference translation, but considering that current metrics are all based on shallow morphological similarity of the MT outputs and the reference, without really understanding the meaning in both sides, the source sentences could have some useful information in differentiating the MT outputs. Consider the Chinese-English translation example below:

Source:	*wo bu neng zhe me zuo*
Hypothesis:	*I must hardly not do this*
Reference:	*I must not do this*

It is clear that the word *not* in the MT output cannot co-exist with the word *hardly* while maintaining the meaning of the source sentence. None of the metrics mentioned above can prevent *not* from being counted in the evaluation, due to the simple reason that they only compute shallow morphological similarity. Then how could the source sentence help in the example? If we reveal the alignment of the source sentence with both the reference and the MT output, the Chinese word *bu neng* would be aligned to *must not* in the reference and *must hardly* in the MT output respectively, leaving the word *not* in the MT output not aligned to any word in the source sentence. Therefore, if we can somehow find the alignments between the source sentence and the reference/MT output, we could be smarter in selecting the overlapping words to be counted in the metric: only select the words which are aligned to the same source words. Now the question comes to how to find the alignment of source sentence and MT hypothesis/references, since the evaluation data set usually does not contain alignment information. Our approach uses GIZA++[2] to construct the many-to-one alignments between source sentences and the MT hypothesis/references respectively.[3] GIZA++ could generate many-to-one alignments either from source sentence to the MT hypothesis, in which case every word in MT hypothesis is aligned to a set of (or none) words in the source sentence, or from the reverse direction, in which case every word in MT hypothesis is aligned to exactly one word (or none) word in the source sentence. In either case, using $MTalign_i$ and $REFalign_i$ to denote the positions of the words in the source sentences which are aligned to a word in the MT hypothesis and a word in the reference respectively, the algorithm for computing source-sentence constrained n-gram precision of length n is described in Figure 1.

```
for all n-grams $w_i, ..., w_{i+n-1}$ in MT hypothesis do
  $max\_val = 0$;
  for all reference sentences do
    for all n-grams $r_j, ..., r_{j+n-1}$ in current reference sentence do
      val=0;
      for k=0; k $\leq$ n-1; k ++ do
        if $w_{i+k}$ equals $r_{j+k}$ AND $MTalign_i$ equals $REFalign_j$ then
          $val$ += $\frac{1}{n}$;
      if $val \geq max\_val$ then
        $max\_val = val$;
  $hit\_count$ += $max\_val$;
return $\frac{hit\_count}{MThypothesislength} \times length\_penalty$;
```

Figure 1: Algorithm for Computing Source-sentence Constrained n-gram Precision

Since source-sentence constrained n-gram precision (SSCN) is a precision-based metric, the vari-

[1]In metrics such as METEOR, ROUGE, SIA (Liu and Gildea, 2006), the positions of words do make difference, but it has nothing to do with the word itself.

[2]GIZA++ is available at http://www.fjoch.com/GIZA++.html

[3]More refined alignments could be got for source-hypothesis from the MT system, and for source-references by using manual proof-reading after the automatic alignment. Doing so, however, requires the MT system's cooperation and some costly human labor.

able $length\_penalty$ is used to avoid assigning a short MT hypothesis a high score, and is computed in the same way as BLEU. Note that in the algorithm for computing the precision of n-grams longer than one word, not all words in the n-grams should satisfy the source-sentence constraint. The reason is that the high order n-grams are already very sparse in the sentence-level evaluation. To differentiate the SSCNs based on the source-to-MT/Ref (many-to-one) alignments and the MT/Ref-to-source (many-to-one) alignments, we use SSCN1 and SSCN2 to denote them respectively. Naturally, we could combine the constraint in SSCN1 and SSCN2 by either taking their union (the combined constrained is satisfied if either one is satisfied) or intersecting them (the combined constrained is satisfied if both constraints are satisfied). We use SSCN_u and SSCN_i to denote the SSCN based on unioned constraints and intersected constraints respectively. We could also apply the stochastic word mapping proposed in SIA (Liu and Gildea, 2006) to replace the hard word matching in Figure 1, and the corresponding metrics are denoted as pSSCN1, pSSCN2, pSSCN_u, pSSCN_i, with the suffixed number denoting different constraints.

2.2 Metrics Based on Source Word Reordering

Most previous MT metrics concentrate on the co-occurrence of the MT hypothesis words in the references. Our metrics based on source sentence reorderings, on the contrary, do not take words identities into account, but rather compute how similarly the source words are reordered in the MT output and the references. For simplicity, we only consider the pairwise reordering similarity. That is, for the source word pair w_i and w_j, if their aligned positions in the MT hypothesis and a reference are in the same order, we call it a consistent word pair. Our pairwise reordering similarity (PRS) metric computes the fraction of the consistent word pairs in the source sentence. Figure 2 gives the formal description of PRS. $SrcMT_i$ and $SrcRef_{k,i}$ denote the aligned position of source word w_i in the MT hypothesis and the kth reference respectively, and N denotes the length of the source sentence.

Another criterion for evaluating the reordering of the source sentence in the MT hypothesis is how well it maintains the original word order in the source sentence. We know that most of the time, the alignment of the source sentence and the MT hypothesis is monotonic. This idea leads to the metric of monotonic pairwise ratio (MPR), which computes the fraction of the source word pairs whose aligned positions in the MT hypothesis are of the same order. It is described in Figure 3.

```
for all word pair $w_i, w_j$ in the source sentence such that $i < j$ do
   for all reference sentences $r_k$ do
      if ($SrcMT_i == SrcMT_j$ AND $SrcRef_{k,i} == SrcRef_{k,j}$) OR (($SrcMT_i - SrcMT_j$) × ($SrcRef_{k,i} - SrcRef_{k,j}$) > 0) then
         $count++$; break;
return $\frac{2 \times count}{N \times (N-1)}$;
```

Figure 2: Compute Pairwise Reordering Similarity

```
for all word pair $w_i, w_j$ in the source sentence, such that $i < j$ do
   if $SrcMT_i - SrcMT_j < 0$ then
      $count++$;
return $\frac{2 \times count}{N \times (N-1)}$;
```

Figure 3: Compute Source Sentence Monotonic Reordering Ratio

2.3 Discriminative Unigram Precision Based on POS

The Discriminative Unigram Precision Based on POS (DUPP) decomposes the normal unigram precision into many sub-precisions according to their POS. The algorithm is described in Figure 4.

These sub-precisions by themselves carry the same information as standard unigram precision, but they provide us the opportunity to make a better combined metric than the normal unigram precision with MCT, which will be introduced in next section.

```
for all unigram s in the MT hypothesis do
   if s is found in any of the references then
      $count_{POS(s)}$ += 1
$precision_x = \frac{count_x}{mt\_hypothesis\_length}$
                $\forall x \in POS$
```

Figure 4: Compute DUPP for N-gram with length n

Such division could in theory be generalized to work with higher order n-grams, but doing so would make the n-grams in each POS set much more sparse. The preprocessing step for the metric is tagging both the MT hypothesis and the references with POS. It might elicit some worries about the robustness of the POS tagger on the noise-containing MT hypothesis. This should not be a problem for two reasons. First, compared with other preprocessing steps like parsing, POS tagging is easier and has higher accuracy. Second, because the counts for each POS are accumulated, the correctness of a single word's POS will not affect the result very much.

3 Maximum Correlation Training for Machine Translation Evaluation

Maximum Correlation Training (MCT) is an instance of the general approach of directly optimizing the objective function by which a model will ultimately be evaluated. In our case, the model is the linear combination of the component metrics, the parameters are the weights for each component metric, and the objective function is the Pearson's correlation of the combined metric and the human judgments. The reason to use the linear combination of the metrics is that the component metrics are usually of the same or similar order of magnitude, and it makes the optimization problem easy to solve. Using w to denote the weights, and m to denote the component metrics, the combined metric x is computed as:

$$x(w) = \sum_j w_j m_j \qquad (1)$$

Using h_i and $x(w)_i$ denote the human judgment and combined metric for a sentence respectively, and N denote the number of sentences in the evaluation set, the objective function is then computed as:

$$Pearson(X(w), H) = \frac{\sum_{i=1}^N x(w)_i h_i - \frac{\sum_{i=1}^N x(w)_i \sum_{i=1}^N h_i}{N}}{\sqrt{(\sum_{i=1}^N x(w)_i^2 - \frac{(\sum_{i=1}^N x(w)_i)^2}{N})(\sum_{i=1}^N h_i^2 - \frac{(\sum_{i=1}^N h_i)^2}{N})}}$$

Now our task is to find the weights for each component metric so that the correlation of the combined metric with the human judgment is maximized. It can be formulated as:

$$w = \operatorname*{argmax}_w Pearson(X(w), H) \qquad (2)$$

The function $Pearson(X(w), H)$ is differentiable with respect to the vector w, and we compute this derivative analytically and perform gradient ascent. Our objective function not always convex (one can easily create a non-convex function by setting the human judgments and individual metrics to some particular value). Thus there is no guarantee that, starting from a random w, we will get the globally optimal w using optimization techniques such as gradient ascent. The easiest way to avoid ending up with a bad local optimum to run gradient ascent by starting from different random points. In our experiments, the difference in each run is very small, i.e., by starting from different random initial values of w, we end up with, not the same, but very similar values for Pearson's correlation.

4 Experiments

Experiments were conducted to evaluate the performance of the new metrics proposed in this paper, as well as the MCT combination framework. The data for the experiments are from the MT evaluation workshop at ACL05. There are seven sets of MT outputs (E09 E11 E12 E14 E15 E17 E22), each of which contains 919 English sentences translated from the same set of Chinese sentences. There are four references (E01, E02, E03, E04) and two sets of human scores for each MT hypothesis. Each human score set contains a fluency and an adequacy score, both of which range from 1 to 5. We create a set of overall human scores by averaging the human fluency and adequacy scores. For evaluating the automatic metrics, we compute the Pearson's correlation of the automatic scores and the averaged human scores (over the two sets of available human scores), for overall score, fluency, and adequacy. The alignment between the source sentences and the MT hypothesis/references is computed by GIZA++, which is trained on the combined corpus of the evaluation data and a parallel corpus of Chinese-English newswire text. The parallel newswire corpus contains around 75,000 sentence pairs, 2,600,000 English words and 2,200,000 Chinese words. The stochastic word mapping is trained on a French-English parallel corpus containing 700,000 sentence pairs, and, following Liu and Gildea (2005), we only keep the top 100 most similar words for each English word.

4.1 Performance of the Individual Metrics

To evaluate our source-sentence based metrics, they are used to evaluate the 7 MT outputs, with the 4 sets of human references. The sentence-level Pearson's correlation with human judgment is computed for each MT output, and the averaged results are shown in Table 1. As a comparison, we also show the results of BLEU, NIST, METEOR, ROUGE, WER, and HWCM. For METEOR and ROUGE, WORDNET and PORTER-STEMMER are enabled, and for SIA, the decay factor is set to 0.6. The number in brackets, for BLEU, shows the n-gram length it counts up to, and for SSCN, shows the length of the n-gram it uses. In the table, the top 3 results in each column are marked bold and the best result is also underlined. The results show that the SSCN2 metrics are better than the SSCN1 metrics in adequacy and overall score. This is understandable since what SSCN metrics need is which words in the source sentence are aligned to an n-gram in the MT hypothesis/references. This is directly modeled in the alignment used in SSCN2. Though we could also get such information from the reverse alignment, as in SSCN1, it is rather an indirect way and could contain more noise. It is interesting that SSCN1 gets better fluency evaluation results than SSCN2. The SSCN metrics with the unioned constraint, SSCN_u, by combining the strength of SSCN1 and SSCN2, get even better results in all three aspects. We can see that SSCN metrics, even without stochastic word mapping, get significantly better results than their relatives, BLEU, which indicates the source sentence constraints do make a difference. SSCN2 and SSCN_u are also competitive to the state-of-art MT metrics such as METEOR and SIA. The best SSCN metric, pSSCN_u(2), achieves the best performance among all the testing metrics in overall and adequacy, and the second best performance in fluency, which is just a little bit worse than the best fluency metric SIA.

	Fluency	Adequacy	Overall
ROUGE_W	24.8	27.8	29.0
ROUGE_S	19.7	30.9	28.5
METEOR	24.4	**34.8**	**33.1**
SIA	**26.8**	32.1	32.6
NIST_1	09.6	22.6	18.5
WER	22.5	27.5	27.7
PRS	14.2	19.4	18.7
MPR	11.0	18.2	16.5
BLEU(1)	18.4	29.6	27.0
BLEU(2)	20.4	31.1	28.9
BLEU(3)	20.7	30.4	28.6
HWCM(2)	22.1	30.3	29.2
SSCN1(1)	24.2	29.6	29.8
SSCN2(1)	22.9	33.0	31.3
SSCN_u(1)	23.8	34.2	32.5
SSCN_i(1)	23.4	28.0	28.5
pSSCN1(1)	24.9	30.2	30.6
pSSCN2(1)	23.8	34.0	32.4
pSSCN_u(1)	24.5	**34.6**	**33.1**
pSSCN_i(1)	24.1	28.8	29.3
SSCN1(2)	24.0	29.6	29.7
SSCN2(2)	23.3	31.5	31.8
SSCN_u(2)	24.1	34.5	**32.8**
SSCN_i(2)	23.1	27.8	28.2
pSSCN1(2)	**24.9**	30.2	30.6
pSSCN2(2)	24.3	34.4	**32.8**
pSSCN_u(2)	**25.2**	**35.4**	**33.9**
pSSCN_i(2)	23.9	28.7	29.1

Table 1: Performance of Component Metrics

The two reordering based metrics, PRS and MPR, are not as good as the other testing metrics, in terms of the individual performance. It should not be surprising since they are totally different kind of metrics, which do not count the overlapping n-grams, but the consistent/monotonic word pair reorderings. As long as they capture some property of the MT hypothesis, they might be able to boost the performance of the combined metric under the MCT framework.

4.2 Performance of the Combined Metrics

To test how well MCT works, the following scheme is used: each set of MT outputs is evaluated by MCT, which is trained on the other 6 sets of MT outputs and their corresponding human judgment; the averaged correlation of the 7 sets of MT outputs with the human judgment is taken as the final result.

4.2.1 Discriminative Unigram Precision based on POS

We first use MCT to combine the discriminative unigram precisions. To reduce the sparseness of the unigrams of each POS, we do not use the original POS set, but use a generalized one by combining all POS tags with the same first letter (e.g., the different verb forms such as *VBN*, *VBD*, and *VBZ* are transformed to *V*). The unified POS set contains 23 POS tags. To give a fair comparison of DUPP with BLEU, the length penalty is also added into it as a component. Results are shown in Table 2. DUPP_f, DUPP_a and DUPP_o denote DUPP trained on human fluency, adequacy and overall judgment respectively. This shows that DUPP achieves obvious improvement over BLEU, with only the unigrams and length penalty, and DUPP_f/_a/_o gets the best result in fluency/adequacy/overall evaluation, showing that MCT is able to make a fluency- or adequacy-oriented metric.

4.2.2 Putting It All Together

The most interesting question in this paper is, with all these metrics, how well we can do in the MT evaluation. To answer the question, we put all the metrics described into the MCT framework and use the combined metric to evaluate the 7 MT outputs. Note that to speed up the training process, we do not directly use 24 DUPP components, instead, we use the 3 combined DUPP metrics. With the metrics shown in Table 1, we then have in total 31 metrics. Table 2 shows the results of the final combined metric. We can see that MCT trained on fluency, adequacy and overall human judgment get the best results among all the testing metrics in fluency, adequacy and overall evaluation respectively. We did a t-test with Fisher's z transform for the combined results and the individual results to see how significant the difference is. The combined results in adequacy and overall are significantly better at 99.5% confidence than the best results of the individual metrics (pSSCN_u(2)), and the combined result in fluency is significantly better at 96.9% confidence than the best individual metric (SIA). We also give the upper bound for each evaluation aspect by training MCT on the testing MT outputs, e.g., we train MCT on E09 and then use it to evaluate E09. The upper-bound is the best we can do with the MCT based on linear combination. Another linear framework, Classification SVM (CSVM),[4] is also used to combine the testing metrics except DUPP. Since DUPP is based on MCT, to make a neat comparison, we rule out DUPP in the experiments with CSVM. The testing scheme is the same as MCT, except that we only use 3 references for each MT hypothesis, and the positive samples for training CSVM are computed as the scores of one of the 4 references based on the other 3 references. The slack parameter of CSVM is chosen so as to maximize the classification accuracy of a heldout set of 800 negative and 800 positive samples, which are randomly selected from the training set. The results are shown in Table 2. We can see that MCT, with the same number of reference sentences, is better than CSVM. Note that the resources required by MCT and CSVM are different. MCT uses human judgments to adjust the weights, while CSVM needs extra human references to produce positive training samples.

	Fluency	Adequacy	Overall
DUPP_f	**23.6**	30.1	30.1
DUPP_a	22.1	**32.9**	30.9
DUPP_o	23.2	32.8	**31.3**
MCT_f(4)	**30.3**	36.7	37.2
MCT_a(4)	28.0	**38.9**	37.4
MCT_o(4)	29.4	38.8	**38.0**
Upper bound	35.3	43.4	42.2
MCT_f(3)	**29.2**	34.7	35.3
MCT_a(3)	27.4	**38.4**	36.8
MCT_o(3)	28.8	38.0	**37.2**
CSVM(3)	27.3	36.9	35.5

Table 2: Combination of the Testing Metrics

To have a rough idea of how the component metrics contribute to the final performance of MCT, we incrementally add metrics into the MCT in descending order of their overall evaluation performance, with the results shown in Figure 5. We can see that the performance improves as the number of metrics increases, in a rough sense. The major improvement happens in the 3rd, 4th, 9th, 14th, and 30th metrics, which are METEOR, SIA, DUPP_a, pSSCN1(1), and PRS. It is interesting to note that these are not the metrics with the highest individual performance. Another interesting observation is that there are no two metrics belonging to the same series in the most beneficial metrics, indicating that to get better combined metrics, individual metrics showing different sentence properties are preferred.

5 Conclusion

This paper first describes two types of new approaches to MT evaluation, which includes making

[4]http://svmlight.joachims.org/

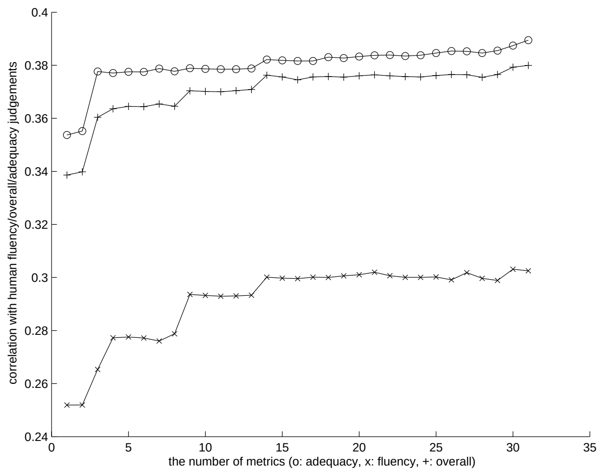


Figure 5: Performance as a Function of the Number of Interpolated Metrics

use of source sentences, and discriminating unigram precisions based on POS. Among all the testing metrics including BLEU, NIST, METEOR, ROUGE, and SIA, our new metric, pSSCN_u(2), based on source-sentence constrained bigrams, achieves the best adequacy and overall evaluation results, and the second best result in fluency evaluation. We further improve the performance by combining the individual metrics under the MCT framework, which is shown to be better than a classification based framework such as SVM. By examining the contribution of each component metric, we find that metrics showing different properties of a sentence are more likely to make a good combined metric.

Acknowledgments This work was supported by NSF grants IIS-0546554, IIS-0428020, and IIS-0325646.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judegments. In *Proceedings of the ACL-04 workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2003. Confidence estimation for machine translation. Technical report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore. Summer Workshop Final Report.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *In HLT 2002, Human Language Technology Conference*, San Diego, CA.

Alex Kulesza and Stuart M. Shieber. 2004. A learning approach to improving sentence-level MT evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Baltimore, MD, October.

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 42th Annual Conference of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain.

Lucian Vlad Lita, Monica Rogati, and Alon Lavie. 2005. Blanc: Learning evaluation metrics for mt. Vancouver.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Ding Liu and Daniel Gildea. 2006. Stochastic iterative alignment for machine translation evaluation. Sydney.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL-03*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL-02*, Philadelphia, PA.

Generating Case Markers in Machine Translation

Kristina Toutanova **Hisami Suzuki**
Microsoft Research
One Microsoft Way, Redmond WA 98052 USA
{hisamis,kristout}@microsoft.com

Abstract

We study the use of rich syntax-based statistical models for generating grammatical case for the purpose of machine translation from a language which does not indicate case explicitly (English) to a language with a rich system of surface case markers (Japanese). We propose an extension of *n*-best re-ranking as a method of integrating such models into a statistical MT system and show that this method substantially outperforms standard *n*-best re-ranking. Our best performing model achieves a statistically significant improvement over the baseline MT system according to the BLEU metric. Human evaluation also confirms the results.

1 Introduction

Generation of grammatical elements such as inflectional endings and case markers is an important component technology for machine translation (MT). Statistical machine translation (SMT) systems, however, have not yet successfully incorporated components that generate grammatical elements in the target language. Most state-of-the-art SMT systems treat grammatical elements in exactly the same way as content words, and rely on general-purpose phrasal translations and target language models to generate these elements (e.g., Och and Ney, 2002; Koehn et al., 2003; Quirk et al., 2005; Chiang, 2005; Galley et al., 2006). However, since these grammatical elements in the target language often correspond to long-range dependencies and/or do not have any words corresponding in the source, they may be difficult to model, and the output of an SMT system is often ungrammatical.

For example, Figure 1 shows an output from our baseline English-to-Japanese SMT system on a sentence from a computer domain. The SMT system, trained on this domain, produces a natural lexical translation for the English word *patch* as *correction program*, and translates *replace* into passive voice, which is more appropriate in Japanese.[1] However, there is a problem in the case marker assignment: the accusative marker *wo*, which was output by the SMT system, is completely inappropriate when the main verb is passive. This type of mistake in case marker assignment is by no means isolated in our SMT system: a manual analysis showed that 16 out of 100 translations had mistakes solely in the assignment of case markers. A better model of case assignment could therefore improve the quality of an SMT system significantly.

S: The patch replaces the .dll file.

O: 修正プログラムを.dll ファイルが置き換えられます。
shuusei puroguramu-wo .dll fairu-ga okikae-raremasu
correction program-ACC dll file-NOM replace-PASS

C: 修正プログラムで.dll ファイルが置き換えられます。
shuusei puroguramu-de .dll fairu-ga okikae-raremasu
correction program-with dll file-NOM replace-PASS

Figure 1: Example of SMT (S: source; O: output of MT; C: correct translation)

In this paper, we explore the use of a statistical model for case marker generation in English-to-Japanese SMT. Though we focus on the generation of case markers in this paper, there are many other surface grammatical phenomena that can be modeled in a similar way, so any SMT system dealing with morpho-syntactically divergent language pairs may benefit from a similar approach to modeling grammatical elements. Our model uses a rich set of syntactic features of both the source (English) and the target (Japanese) sentences, using context which is broader than that utilized by existing SMT systems. We show that the use of such features results in very high case assignment quality and also leads to a notable improvement in MT quality.

Previous work has discussed the building of special-purpose classifiers which generate grammatical elements such as prepositions (Hajič et al. 2002), determiners (Knight and Chander, 1994) and case markers (Suzuki and Toutanova, 2006) with an eye toward improving MT output. How-

[1] There is a strong tendency to avoid transitive sentences with an inanimate subject in Japanese.

ever, these components have not actually been integrated in an MT system. To our knowledge, this is the first work to integrate a grammatical element production model in an SMT system and to evaluate its impact in the context of end-to-end MT.

A common approach of integrating new models with a statistical MT system is to add them as new feature functions which are used in decoding or in models which re-rank *n*-best lists from the MT system (Och et al., 2004). In this paper we propose an extension of the *n*-best re-ranking approach, where we expand *n*-best candidate lists with multiple case assignment variations, and define new feature functions on this expanded candidate set. We show that expanding the *n*-best lists significantly outperforms standard *n*-best re-ranking. We also show that integrating our case prediction model improves the quality of translation according to BLEU (Papineni et al., 2002) and human evaluation.

2 Background

In this section, we provide necessary background of the current work.

2.1 Task of case marker prediction

Our definition of the case marker prediction task follows Suzuki and Toutanova (2006). That is, we assume that we are given a source English sentence, and its translation in Japanese which does not include case markers. Our task is to predict all case markers in the Japanese sentence.

We determine the location of case marker insertion using the notion of *bunsetsu*. A bunsetsu consists of one content (head) word followed by any number of function words. We can therefore segment any sentence into a sequence of bunsetsu by using a part-of-speech (POS) tagger.

Once a sentence is segmented into bunsetsu, it is trivial to determine the location of case markers in a sentence: each bunsetsu can have at most one case marker, and the position of the case maker within a phrase is predictable, i.e., the rightmost position before any punctuation marks. The sentence in Figure 1 thus has the following bunsetsu analysis (denoted by square brackets), with the locations of potential case marker insertion indicated by □:

[修正'correction'□] [プログラム'program'□] [.dll□] [ファイル'file'□] [置き換えられます'replace-PASS'□。]

For each of these positions, our task is to predict the case marker or to predict NONE, which means that the phrase does not have a case marker.

case markers		grammatical functions	+*wa*
が	*ga*	subject; object	
を	*wo*	object; path	
の	*no*	genitive; subject	
に	*ni*	dative object, location	✓
から	*kara*	source	✓
と	*to*	quotative, reciprocal, *as*	✓
で	*de*	location,instrument, cause	✓
へ	*e*	goal, direction	✓
まで	*made*	goal (up to, until)	✓
より	*yori*	source, comparison target	✓
は	*wa*	Topic	

Table 1. Case markers to be predicted

The case markers we used for the prediction task are the same as those defined in Suzuki and Toutatnova (2006), and are summarized in Table 1: in addition to the case markers in a strict sense, the topic marker *wa* is also included as well as the combination of a case marker plus the topic marker for the case markers with the column +*wa* checked in the table. In total, there are 18 case markers to predict: ten simple case markers, the topic marker *wa*, and seven case+*wa* combinations. The case prediction task is therefore a 19-fold classification task: for each phrase, we assign one of the 18 case markers or NONE.

2.2 Treelet translation system

We constructed and evaluated our case prediction model in the context of a treelet-based translation system, described in Quirk et al. (2005).[2] In this approach, translation is guided by treelet translation pairs, where a treelet is a connected subgraph of a dependency tree.

A sentence is translated in the treelet system as follows. The input sentence is first parsed into a dependency structure, which is then partitioned into treelets, assuming a uniform probability distribution over all partitions. Each source treelet is then matched to a treelet translation pair, the collection of which will form the target translation. The target language treelets are then joined to form a single tree, and the ordering of all the nodes is determined, using the method described in Quirk et al. (2005).

Translations are scored according to a linear combination of feature functions:

$$score(t) = \sum_j \lambda_j f_j(t) \qquad (1)$$

[2] Though this paper reports results in the context of a treelet system, the model is also applicable to other syntax-based or phrase-based SMT systems.

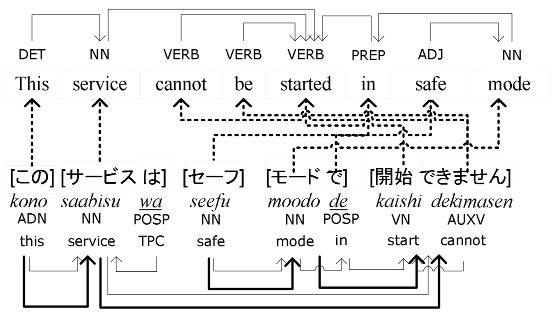


Figure 2. Aligned English-Japanese sentence pair

where λ_j are the model parameters and $f_j(t)$ is the value of the feature function j on the candidate t. There are ten feature functions in the treelet system, including log-probabilities according to inverted and direct channel models estimated by relative frequency, lexical weighting channel models following Vogel et al. (2003), a trigram target language model, an order model, word count, phrase count, average phrase size functions, and whole-sentence IBM Model 1 log-probabilities in both directions (Och et al. 2004). The weights of these models are determined using the max-BLEU method described in Och (2003). As we describe in Section 4, the case prediction model is integrated into the system as an additional feature function.

The treelet translation model is estimated using a parallel corpus. First, the corpus is word-aligned using GIZA++ (Och and Ney, 2000); then the source sentences are parsed into a dependency structure, and the dependency is projected onto the target side following the heuristics described in Quirk et al. (2005). Figure 2 shows an example of an aligned sentence pair: on the source (English) side, POS tags and word dependency structure are assigned (solid arcs); the word alignments between English and Japanese words are indicated by the dotted lines. On the target (Japanese) side, projected word dependencies (solid arcs) are available. Additional annotations in Figure 2, namely the POS tags and the bunsetsu dependency structure (bold arcs) on the target side, are derived from the treelet system to be used for building a case prediction model, which we describe in Section 3.

2.3 Data

All experiments reported in this paper are run using parallel data from a technical (computer) domain. We used two main data sets: train-500K, consisting of 500K sentence pairs which we used for training the baseline treelet system as well as the case prediction model, and a disjoint set of three data sets, lambda-1K, dev-1K and test-2K, which are used to integrate and evaluate the case prediction model in an end-to-end MT scenario. Some characteristics of these data sets are given in Table 2. We will refer to this table as we describe our experiments in later sections.

data set	# sent pairs	# of words (average sent length in words)	
		English	Japanese
train-500K	500K	7,909,198 (15.81)	9,379,240 (18.75)
lambda-1K	1,000	15,219(15.2)	20,660 (20.7)
dev-1K	1,000	15,397(15.4)	21,280 (21.3)
test-2K	2,000	30,198(15.1)	41,269 (20.6)

Table 2: Data set characteristics

3 Statistical Models for Case Prediction in MT

3.1 Case prediction model

Our model of case marker prediction closely follows our previous work of case prediction in a non-MT context (Suzuki and Toutanova, 2006). The model is a multi-class log-linear (maximum entropy) classifier using 19 classes (18 case markers and NONE). It assigns a probability distribution over case marker assignments given a source English sentence, all non-case marker words of a candidate Japanese translation, and additional annotation information. Let t denote a Japanese translation, s a corresponding source sentence, and A additional annotation information such as alignment, dependency structure, and POS tags (such as shown in Figure 2). Let $rest(t)$ denote the sequence of words in t excluding all case markers, and $case(t)$ a case marking assignment for all phrases in t. Our case marking model estimates the probability of a case assignment given all other information:

$$P_{case}\,(case\,(t)\,|\,rest\,(t),\,s,\,A)$$

The probability of a complete case assignment is a product over all phrases of the probability of the case marker of the phrase given all context features used by the model. Our model assumes that the case markers in a sentence are independent of each other given the input features. This independence assumption may seem strong, but the results presented in our previous work (Suzuki and Toutanova, 2006) showed that a joint model did not result in large improvements over a local one in predicting case markers in a non-MT context.

3.2 Model features and feature selection

The features of our model are similar to the ones described in Suzuki and Toutanova (2006). The main difference is that in the current model we applied a feature selection and induction algorithm to determine the most useful features and feature combinations. This is important for understanding what sources of information are important for predicting grammatical elements, but are currently absent from SMT systems. We used 490K sentence pairs for training the case prediction model, which is a subset of the train-500K set of Table 2. We divided the remaining 10K sentences for feature selection (5K-feat) and for evaluating the case prediction models on reference translations (5K-test, discussed in Section 3.3). The paired data is annotated using the treelet translation system: as shown in Figure 2, we have source and target word dependency structure, source language POS and word alignment directly from the aligned treelet structure. Additionally, we used a POS tagger of Japanese to assign POS to the target sentence as well as to parse the sentence into bunsetsu (indicated by brackets in Figure 2), using the method described in Section 2.1. We then compute bunsetsu dependency structure on the target side (indicated by bold arcs in Figure 2) based on the word dependency structure projected from English. We apply this procedure to annotate a paired corpus (in which case the Japanese sentence is a reference translation) as well as translations generated by the SMT system (which may potentially be ill-formed).

We derived a large set of possible features from these annotations. The features are represented as feature templates, such as "Headword POS=X", which generate a set of binary features corresponding to different instantiations of the template, such as "Headword POS=NOUN". We applied an automatic feature selection and induction algorithm to the base set of templates.

The feature selection algorithm considers the original templates as well as arbitrary (bigram and trigram) conjunctions of these templates. The algorithm performs forward stepwise feature selection, choosing templates which result in the highest increase in model accuracy on the 5K-feat set mentioned above. The algorithm is similar to the one described in McCallum (2003).

The application of this feature selection procedure gave us 17 templates, some of which are shown in Table 3, along with example instantiations for the phrase headed by *saabisu* 'service' from Figure 2. Conjunctions are indicated by &. Note that many features that refer to POS and syntactic (parent) information are selected, on both the target and source sides. We also note that the context required by these features is more extensive than what is usually available during decoding in an SMT system due to a limit imposed on the treelet or phrase size. For example, our model uses word lemma and POS tags of up to six words (previous word, next word, word in position +2, head word, previous head word and parent word), which covers more context than the treelet system we used (the system imposes the treelet size limit of four words). This means that the case model can make use of much richer information from both the source and target than the baseline MT system. Furthermore, our model makes better use of the context by combining the contributions of multiple sources of knowledge using a maximum entropy model, rather than using the relative frequency estimates with a very limited amount of smoothing, which are used by most state-of-the art SMT systems.

Features	Example
Words in position −1 and +2	*kono,moodo*
Headword & previous headword	*saabisu&kono*
Parent word	*kaishi*
Aligned word	*service*
Parent of word aligned to headword	*started*
Next word POS	*NOUN*
Next word & next word POS	*seefu&NN*
Headword POS	*NOUN*
Parent headword POS	*VN*
Aligned to parent word POS & next word POS & prev word POS	*VERB&NN&and*
Parent POS of word aligned to headword	*VERB*
Aligned word POS & headword POS & prev word POS	*NN&NN&ADN*
POS of word aligned to headword	*NOUN*

Table 3: Features for the case prediction model

3.3 Performance on reference translations

Before discussing the integration of the case prediction model with the MT system, we present an evaluation of the model on the task of predicting the case assignment of *reference* translations. This performance constitutes an upper bound on the model's performance in MT, because in reference translations, the word choice and the word order are perfect.

Table 4 summarizes the results of the reference experiments on the 5K-test set using two metrics: accuracy, which denotes the percentage of phrases for which the respective model guessed the case marker correctly, and BLEU score against the reference translation. For com-

Model	ACC	BLEU
Baseline (frequency)	58.9	40.0
Baseline (490K LM)	87.2	83.6
Log-linear model	94.9	93.0

Table 4: Accuracy (%) and BLEU score for case prediction when given correct context (reference translations) on the 5K-test set

parison, we also include results from two baselines: a frequency-based baseline, which always assigns the most likely class (NONE), and a language model (LM) baseline, which is one of the standard methods of generating grammatical elements in MT. We trained a word-trigram LM using the CMU toolkit (Clarkson and Rosenfeld, 1997) on the same 490K sentences which we used for training the case prediction model.

Table 4 shows that our model performs substantially better than both baselines: the accuracy of the frequency-based baseline is 59%, and an LM-based model improves it to 87.2%. In contrast, our model achieves an accuracy of 95%, which is a 60% error reduction over the LM baseline. It is also interesting to note that as the accuracy goes up, so does the BLEU score.

These results show that our best model can very effectively predict case markers when the input to the model is clean, i.e., when the input has correct words in correct order. Next, we see the impact of applying this model to improve MT output.

4 Integrating Case Prediction Models in MT

In the end-to-end MT scenario, we integrate our case assignment model with the SMT system and evaluate its contribution to the final MT output.

As a method of integration with the MT system, we chose an *n*-best re-ranking approach, where the baseline MT system is left unchanged and additional models are integrated in the form of feature functions via re-ranking of *n*-best lists from the system. Such an approach has been taken by Och et al. (2004) for integrating sophisticated syntax-informed models in a phrase-based SMT system. We also chose this approach for ease of implementation: as discussed in Section 3.2, the features we use in our case model extend over long distance, and are not readily available during decoding. Though a tighter integration with the decoding process is certainly worth exploring in the future, we have taken an approach here that allows fast experimentation.

Within the space of *n*-best re-ranking, we have considered two variations: the standard *n*-best re-ranking method, and our significantly better performing extension. These are now discussed in turn.

4.1 Method 1: Standard *n*-best re-ranking

This method is a straightforward application of the *n*-best re-ranking approach described in Och et al. (2004). As described in Section 2.2, our baseline SMT system is a linear model which weighs the values of ten feature functions. To integrate a case prediction model, we simply add it to the linear model as an 11th feature function, whose value is the log-probability of the case assignment of the candidate hypothesis *t* according to our model. The weights of all feature functions are then re-estimated using max-BLEU training on the *n*-best list of the lambda-1K set in Table 2. As we show in Section 5, this re-ranking method did not result in good performance.

4.2 Method 2: Re-ranking of expanded candidate lists

A drawback of the previous method is that in an *n*-best list, there may not be sufficiently many case assignment variations of existing hypotheses. If this is the case, the model cannot be effective in choosing a hypothesis with a good case assignment. We performed a simple experiment to test this. We took the first (best) hypothesis *t* from the MT system and generated the top 40 case variations *t'* of *t*, according to the case assignment model. These variations differ from *t* only in their case markers. We wanted to see what fraction of these new hypotheses *t'* occurred in a 1000-best list of the MT system. In the dev-1K set of Table 2, the fraction of new case variations of the first hypothesis occurring in the 1000-best list of hypotheses was 0.023. This means that only less than one (2.3% of 40 = 0.92) case variant of the first hypothesis is expected to be found in the 1000-best list, indicating that even an *n*-best list for a reasonably large *n* (such as 1000) does not contain enough candidates varying in case marker assignment.

In order to allow more case marking candidates to be considered, we propose the following method to expand the candidate translation list: for each translation *t* in the *n*-best list of the baseline SMT system, we also consider case assignment variations of *t*. For simplicity, we chose to consider the top *k* case assignment variations of each hypothesis according to our case model,[3] for $1 \leq k \leq 40$.[4]

[3] From a computational standpoint, it is non-trivial to con-

After we expand the translation candidate set, we compute feature functions for all candidates and train a linear model which chooses from this larger set. While some features (e.g., word count feature) are easy to recompute for a new candidate, other features (e.g., treelet phrase translation probability) are difficult to recompute. We have chosen to recompute only four features of the baseline model: the language model feature, the word count feature, and the direct and reverse whole-sentence IBM Model 1 features, assuming that the values of the other baseline model features for a casing variation *t'* of *t* are the same as their values for *t*. In addition, we added the following four feature functions, specifically meant to capture the extent to which the newly generated case marking variations differ from the original baseline system hypotheses they are derived from:

- **Generated:** a binary feature with a value of 0 for original baseline system candidates, and a value of 1 for newly generated candidates.
- **Number NONE→non-NONE:** the count of case markers changed from NONE to non-NONE with respect to an original translation candidate.
- **Number non-NONE→NONE:** the count of case markers changed from non-NONE to NONE.
- **Number non-NONE→non-NONE:** the count of case markers changed from non-NONE to another non-NONE case marker.

Note that these newly defined features all have a value of 0 for original baseline system candidates (i.e., when *k*=0) and therefore would have no effect in Method 1. Therefore, the only difference between our two methods of integration is the presence or absence of case-expanded candidate translations.

5 Experiments and Results

5.1 Data and settings

For our end-to-end MT experiments, we used three datasets in Table 2 that are disjoint from the train-500K data set. They consist of source English sentences and their top 1000 candidate translations produced by the baseline SMT system. These datasets are the lambda-1K set for training the weights λ of the linear model from Equation (1), the dev-1K set for model selection, and the test-2K set for final testing including human evaluation.

Models	#MT hypotheses	#case expansions	BLEU	Oracle BLEU
Baseline	1	0	37.99	37.99
Method 1	20	0	37.83	41.79
	100	0	38.02	42.79
	1000	0	38.08	43.14
Method 2	1	1	38.18	38.75
	1	10	38.42	40.51
	1	20	38.54	41.15
	1	40	38.41	41.74
Method 2	20	10	**38.91**	45.32
	20	20	38.72	45.94
	20	40	38.78	46.56
	100	10	38.73	46.87
	100	20	38.64	47.47
	100	40	38.74	47.96

Table 5. Results of end-to-end experiments on the dev-1K set

5.2 Results

The results for the end-to-end experiments on the dev-1K set are summarized in Table 5. The table is divided into four sections. The first section (row) shows the BLEU score of the baseline SMT system, which is equivalent to the 1-best re-ranking scenario with no case expansion. The BLEU score for the baseline was 37.99. In the table, we also show the oracle BLEU scores for each model, which are computed by greedily selecting the translation in the candidate list with the highest BLEU score.[5]

The second section of Table 5 corresponds to the results obtained by Method 1, i.e., the standard *n*-best re-ranking, for *n* = 20, 100, and 1000. Even though the oracle scores improve as *n* is increased, the actual performance improves only slightly. These results show that the strategy of only including the new information as features in a standard *n*-best re-ranking scenario does not lead to an improvement over the baseline.

In contrast, Method 2 obtains notable improvements over the baseline. Recall that we expand the *n*-best SMT candidates with their *k*-best case marking variations in this method, and re-

sider all possible case assignment variations of a hypothesis: even though the case assignment score for a sentence is locally decomposable, there are still global dependencies in the linear model from Equation (1) due to the reverse whole-sentence IBM model 1 score used as a feature function.

[4] Our results indicate that additional case variations would not be helpful.

[5] A modified version of BLEU was used to compute sentence-level BLEU in order to select the best hypothesis per sentence. The table shows corpus-level BLEU on the resulting set of translations.

train the model parameters on the resulting candidate lists. For the values *n*=1 and *k*=1 (which we refer to as 1best-1case), we observe a small BLEU gain of .19 over the baseline. Even though this is not a big improvement, it is still better than the improvement of standard *n*-best reranking with a 1000-best list. By considering more case marker variations (*k* = 10, 20 and 40), we are able to gain about a half BLEU point over the baseline. The fact that using more case variations performs better than using only the best case assignment candidate proposed by the case model suggests that the proposed approach, which integrates the case prediction model as a feature function and retrains the weights of the linear model, works better than using the case prediction model as a post-processor of the MT output.

The last section of the table explores combinations of the values for *n* and *k*. Considering 20 best SMT candidates and their top 10 case variations gave the highest BLEU score on the dev-1K set of 38.91, which is an 0.92 BLEU points improvement over the baseline. Considering more case variations (20 or 40), and more SMT candidates (100) resulted in a similar but slightly lower performance in BLEU. This is presumably because the case model does affect the choice of content words as well, but this influence is limited and can be best captured when using a small number (*n*=20) of baseline system candidates.

Based on these results on the dev-1K set, we chose the best model (i.e., 20-best-10case) and evaluated it on the test-2K set against the baseline. Using the pair-wise statistical test design described in Collins et al. (2005), the BLEU improvement (35.53 vs. 36.29) was statistically significant (*p* < .01) according to the Wilcoxon signed-rank test.

5.3 Human evaluation

These results demonstrate that the proposed model is effective at improving the translation quality according to the BLEU score. In this section, we report the results of human evaluation to ensure that the improvements in BLEU lead to better translations according to human evaluators.

We performed human evaluation on the 20best-10case (*n*=20, *k*=10) and 1best-40case (*n*=1, *k*=40) models against the baseline using our final test set, the test-2K data. The performance in BLEU of these models on the full test-2K data was 35.53 for the baseline, 36.09 for the 1best-40case model, and 36.29 for the 20best-10case model, respectively.

		Fluency			Adequacy		
		Annotator #1			Annotator #1		
		S	B	E	S	B	E
Anno-tator #2	S	**27**	1	8	**17**	0	9
	B	1	**9**	16	0	**9**	12
	E	7	4	**27**	9	8	**36**

Table 6. Results of human evaluation comparing 20best-10case vs. baseline. **S**: proposed system is better; **B**: baseline is better; **E**: of equal quality

In our human evaluation, two annotators were asked to evaluate a random set of 100 sentences for which the models being compared produced different translations. The judges were asked to compare two translations, the baseline output from the original SMT system and the output chosen by the system augmented with the case marker generation component. Each judge was asked to run two separate evaluations along different evaluation criteria. In the evaluation of *fluency*, the judges were asked to decide which translation is more readable/grammatical, ignoring the reference translation. In the evaluation of *adequacy*, they were asked to judge which translation more correctly reflects the meaning of the reference translation. In either setting, they were not given the source sentence.

Table 6 summarizes the results of the evaluation of the 20best-10case model. The table shows the results along two evaluation criteria separately, fluency on the left and adequacy on the right. The evaluation results of Annotator #1 are shown in the columns, while those of Annotator #2 are in the rows. Each grid in the table shows the number of sentences the annotators classified as the proposed system output better (S), the baseline system better (B) or the translations are of equal quality (E). Along the diagonal (in boldface) are the judgments that were agreed on by the two annotators: both annotators judged the output of the proposed system to be more fluent in 27 translations, less fluent in 9 translations; they judged that our system output was more adequate in 17 translations and less adequate in 9 translations. Our system output was thus judged better under both criteria, though according to a sign test, the improvement is statistically significant (*p* < .01) in fluency, but not in adequacy.

One of the reasons for this inconclusive result is that human evaluation may be very difficult and can be unreliable when evaluating very different translation candidates, which happens often when comparing the results of models that consider *n*-best candidates where *n*>1, as is the case with the 20best-10case model. In Table 6,

		Fluency			Adequacy		
		Annotator #1			Annotator #1		
		S	B	E	S	B	E
Anno-tator #2	S	**42**	0	9	**30**	1	9
	B	1	**0**	7	0	**9**	7
	E	7	2	**32**	9	3	**32**

Table 7. Results of human evaluation comparing 1best-40case vs. baseline

we can see that the raw agreement rate between the two annotators (i.e., number of agreed judgments over all judgments) is only 63% (27+9+27 /100) in fluency and 62% (17+9+36/100) in adequacy. We therefore performed an additional human evaluation where translations being compared differ only in case markers: the baseline vs. the 1best-40case model output. The results are shown in Table 7.

This evaluation has a higher rate of agreement, 74% for fluency and 71% for adequacy, indicating that comparing two translations that differ only minimally (i.e., in case markers) is more reliable. The improvements achieved by our model are statistically significant in both fluency and adequacy according to a sign test; in particular, it is remarkable that on 42 sentences, the judges agreed that our system was better in fluency, and there were no sentences on which the judges agreed that our system caused degradation. This means that the proposed system, when choosing among candidates differing only in case markers, can improve the quality of MT output in an extremely precise manner, i.e. making improvements without causing degradations.

6 Conclusion

We have described a method of using a case marker generation model to improve the quality of English-to-Japanese MT output. We have shown that the use of such a model contributes to improving MT output, both in BLEU and human evaluation. We have also proposed an extension of *n*-best re-ranking which significantly outperformed standard *n*-best re-ranking. This method should be generally applicable to integrating models which target specific phenomena in translation, and for which an extremely large *n*-best list would be needed to cover enough variants of the phenomena in question.

Our model improves the quality of generated case markers in an extremely precise manner. We believe this result is significant, as there are many phenomena in the target language of MT that may be improved by using special-purpose models, including the generation of articles, auxiliaries, inflection and agreement. We plan to extend and generalize the current approach to cover these phenomena in morphologically complex languages in general in the future.

References

Clarkson, P.R. and R. Rosenfeld. 1997. Statistical Language Modeling Using the CMU-Cambridge Toolkit. In *ESCA Eurospeech*, pp. 2007-2010.

Collins, M., P. Koehn and I. Kučerová. 2005. Clause Restructuring for Statistical Machine Translation. In *ACL*, pp.531-540.

Chiang, D. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *ACL*.

Galley, M., J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang and I. Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *ACL*.

Koehn, P., F. J. Och and D. Marcu. 2003. Statistical Phrase-based Translation. In *HLT-NAACL*.

Hajič, J., M. Čmejrek, B. Dorr, Y. Ding, J. Eisner, D. Gildea, T. Koo, K. Parton, G. Penn, D. Radev and O. Rambow. 2002. Natural Language Generation in the Context of Machine Translation. *Technical report, Center for Language and Speech Processing, Johns Hopkins University 2002 Summer Workshop Final Report.*

Knight, K. and I. Chander. 1994. Automatic Postediting of Documents. In *AAAI*.

McCallum, A. 2003. Efficiently inducing features of conditional random fields. *In UAI.*

Och, F. J. 2003. Minimum Error-rate Training for Statistical Machine Translation. In *ACL*.

Och, F. J. and H. Ney. 2000. Improved Statistical Alignment Models. In *ACL*.

Och, F. J. and H. Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *ACL 2002.*

Och, F. J., D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin and D. Radev. 2004. A Smorgasbord of Features for Statistical Machine Translation. In *NAACL*.

Papineni, K., S. Roukos, T. Ward and W.J. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL*.

Quirk, C., A. Menezes and C. Cherry. 2005. Dependency Tree Translation: Syntactically Informed Phrasal SMT. In *ACL*.

Suzuki, H. and K. Toutanova. 2006. Learning to Predict Case Markers in Japanese. In *ACL-COLING*.

Vogel, S., Y. Zhang, F. Huang, A. Tribble, A. Venugopal, B. Zhao and A. Waibel. 2003. The CMU Statistical Machine Translation System. In *Proceedings of the MT Summit*.

Direct Translation Model 2

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 machine translation system using a minimal set of translation blocks (phrase-pairs). While recent phrase-based statistical machine translation (SMT) systems achieve significant improvement over the original source-channel statistical translation models, they 1) use a *large* inventory of blocks which have significant overlap and 2) limit the use of training to just a few parameters (on the order of ten). In contrast, we show that our proposed minimalist system (DTM2) achieves equal or better performance by 1) recasting the translation problem in the traditional statistical modeling approach using blocks with no overlap and 2) relying on training most system parameters (on the order of millions or larger). The new model is a direct translation model (DTM) formulation which allows easy integration of additional/alternative views of both source and target sentences such as segmentation for a source language such as Arabic, part-of-speech of both source and target, etc. We show improvements over a state-of-the-art phrase-based decoder in Arabic-English translation.

1 Introduction

Statistical machine translation takes a source sequence, $\boldsymbol{S} = [s_1\ s_2\ \ldots\ s_K]$, and generates a target sequence, $\boldsymbol{T}^* = [t_1\ t_2\ \ldots\ t_L]$, by finding the most likely translation given by:

$$\boldsymbol{T}^* = \arg\max_{\boldsymbol{T}} p(\boldsymbol{T}|\boldsymbol{S}).$$

1.1 Block selection

Recent statistical machine translation (SMT) algorithms generate such a translation by incorporating an inventory of bilingual phrases (Och and Ney, 2000). A *m-n* phrase-pair, or block, is a sequence of m source words paired with a sequence of n target words. The inventory of blocks in current systems is highly redundant. We illustrate the redundancy using the example in Table 1 which shows a set of phrases that cover the two-word Arabic fragment "*lljnp Almrkzyp*" whose alignment and translation is shown in Figure 1. One notices the significant overlap between the various blocks including the fact the output target sequence "*of the central committee*" can be produced in at least two different ways: 1) as 2-4 block "*lljnp Almrkzyp | of the central committee*" covering the two Arabic words, or 2) by using the 1-3 block "*Almrkzyp | of the central*" followed by covering the first Arabic word with the 1-1 block "*lljnp | committee*". In addition, if one adds one more word to the Arabic fragment in the third position such as the block "*AlSyny | chinese*" the overlap increases significantly and more alternate possibilities are available to produce an output such as the "*of the central chinese committee.*"

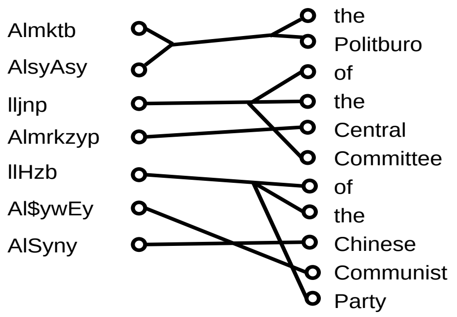


Figure 1: Example of Arabic snipet and alignment to its English translation.

In this work, we propose to only use 1-n blocks and avoid completely the redundancy obtained by the use of m-n blocks for $m > 1$ in current phrase-based systems. We discuss later how by defining appropriate features in the translation model, we capture the important dependencies required for producing n-long fragments for an m-word input sequence including the reordering required to produce more fluent output. So in Table 1 only the blocks corresponding to a single Arabic word are in the block inventory. To differentiate this work from previous approaches in

lljnp	**Almrkzyp**
committee	central
of the commission	the central
commission	of the central
of the committee	of central
the committee	and the central
of the commission on	and central
the commission	, central
committee of	's central
...	...
of the central committee(11)	
of the central committee of (11)	
the central committee of (8)	
central committee(7)	
committee central (2)	
central committee , (2)	
...	

Table 1: Example Arabic-English blocks showing possible 1-n and 2-n blocks ranked by frequency. Block count is given in () for 2-n blocks.

direct modeling for machine translation, we call our current approach DTM2 (Direct Translation Model 2).

1.2 Statistical modeling for translation

Earlier work in statistical machine translation (Brown et al., 1993) is based on the "noisy-channel" formulation where

$$\boldsymbol{T}^* = \arg\max_{\boldsymbol{T}} p(\boldsymbol{T}|\boldsymbol{S}) = \arg\max_{\boldsymbol{T}} p(\boldsymbol{T})p(\boldsymbol{S}|\boldsymbol{T}) \quad (1)$$

where the target language model $p(\boldsymbol{T})$ is further decomposed as

$$p(\boldsymbol{T}) \propto \prod_i p(t_i|t_{i-1}, \ldots, t_{i-k+1})$$

where k is the order of the language model and the translation model $p(\boldsymbol{S}|\boldsymbol{T})$ has been modeled by a sequence of five models with increasing complexity (Brown et al., 1993). The parameters of each of the two components are estimated using Maximum Likelihood Estimation (MLE). The LM is estimated by counting n-grams and using smoothing techniques. The translation model is estimated via the EM algorithm or approximations that are bootstrapped from the previous model in the sequence as introduced in (Brown et al., 1993). As is well known, improved results are achieved by modifying the Bayes factorization in Equation 1 above by weighing each distribution differently as in:

$$p(\boldsymbol{T}|S) \propto p^{\alpha}(\boldsymbol{T})p^{1-\alpha}(\boldsymbol{S}|\boldsymbol{T}) \quad (2)$$

This is the simplest MaxEnt[1] model that uses two feature functions. The parameter α is tuned on a development set (usually to improve an error metric instead of MLE). This model is a special case of the Direct Translation Model proposed in (Papineni et al., 1997; Papineni et al., 1998) for language understanding; (Foster, 2000) demostrated perplexity reductions by using direct models; and (Och and Ney, 2002) employed it very successfully for language translation by using about ten feature functions:

$$p(\boldsymbol{T}|\boldsymbol{S}) = \frac{1}{Z}\exp\sum_i \lambda_i \phi_i(\boldsymbol{S}, \boldsymbol{T})$$

Many of the feature functions used for translation are MLE models (or smoothed variants). For example, if one uses $\phi_1 = log(p(\boldsymbol{T}))$ and $\phi_2 = log(p(\boldsymbol{S}|\boldsymbol{T}))$ we get the model described in Equation 2. Most phrase-based systems, including the baseline decoder used in this work use feature functions:

- a target word n-gram model (e.g., $n = 5$),
- a target part-of-speech n-gram model ($n \geq 5$),
- various translation models such as a block inventory with the following three varieties: 1) the unigram block count, 2) a model 1 score $p(\boldsymbol{s}_i|\boldsymbol{t}_i)$ on the phrase-pair, and 3)a model 1 score for the other direction $p(\boldsymbol{t}_i|\boldsymbol{s}_i)$,
- a target word count penalty feature $|\boldsymbol{T}|$,
- a phrase count feature,
- a distortion model (Al-Onaizan and Papineni, 2006).

The weight vector $\boldsymbol{\lambda}$ is estimated by tuning on a rather *small* (as compared to the training set used to define the feature functions) development set using the BLEU metric (or other translation error metrics). Unlike MaxEnt training, the method (Och, 2003) used for estimating the weight vector for BLEU maximization are not computationally scalable for a large number of feature functions.

2 Related Work

Most recent state-of-the-art machine translation decoders have the following aspects that we improve upon in this work: 1) block style, and 2) model parameterization and parameter estimation. We discuss each item next.

[1]The subfields of log-linear models, exponential family, and MaxEnt describe the equivalent techniques from different perspectives.

2.1 Block style

In order to extract phrases from alignments available in one or both directions, most SMT approaches use a heuristic such as *union*, *intersection*, *inverse projection constraint*, etc. As discussed earlier, these approaches result in a large overlap between the extracted blocks (longer blocks overlap with all the shorter subcomponents blocks). Also, slightly restating the advantages of phrase-pairs identified in (Quirk and Menezes, 2006), these blocks are effective at capturing context including the encoding of non-compositional phrase pairs, and capturing local reordering, but they lack variables (e.g. embedding between $ne \ldots pas$ in French), have sparsity problems, and lack a strategy for global reordering. More recently, (Chiang, 2005) extended phrase-pairs (or blocks) to hierarchical phrase-pairs where a grammar with a single non-terminal allows the embedding of phrases-pairs, to allow for arbitrary embedding and capture global reordering though this approach still has the high overlap problem. However, in (Quirk and Menezes, 2006), the authors investigate minimum translation units (MTU) which is a refinement over a similar approach by (Banchs et al., 2005) to eliminate the overlap issue. The MTU approach picks all the minimal blocks subject to the condition that no word alignment link crosses distinct blocks. They do not have the notion of a block with a variable (a special case of the hierarchical phrase-pairs) that we employ in this work. They also have a weakness in the parameter estimation method; they rely on an n-gram language model on blocks which inherently requires a large bilingual training data set.

2.2 Estimating Model Parameters

Most recent SMT systems use blocks (i.e. phrase-pairs) with a few real valued "informative" features which can be viewed as an indicator of how probable the current translation is. As discussed in Section 1.2, these features are typically MLE models (e.g. block translation, Model 1, language model, etc.) whose scores are log-linearly combined using a weight vector, λ_f where f is a particular feature. The λ_f are trained using a held-out corpus using maximum BLEU training (Och, 2003). This method is only practical for a small number of features; typically, the number of features is on the order of 10 to 20.

Recently, there have been several discriminative approaches at training large parameter sets including (Tillmann and Zhang, 2006) and (Liang et al., 2006). In (Tillmann and Zhang, 2006) the model is optimized to produce a block orientation and the target sentence is used only for computing a sentence level BLEU. (Liang et al., 2006) demonstrates a discriminatively trained system for machine translation that has the following characteristics: 1) requires a varying update strategy (local vs. bold) depending on whether the reference sentence is "reachable" or not, 2) uses sentence level BLEU as a criterion for selecting which output to update towards, and 3) only trains on limited length (5-15 words) sentences.

So both methods fundamentally rely on a prior decoder to produce an "N-best" list that is used to find a target (using max BLEU) for the training algorithm. The methods to produce an "N-best" list tend to be not very effective since most alternative translations are minor differences from the highest scoring translation and do not typically include the reference translation (particularly when the system makes a large error).

In this paper, the algorithm trains on all sentences in the test-specific corpus and crucially, the algorithm directly uses the target translation to update the model parameters. This latter point is a critical difference that contrasts to the major weakness of the work of (Liang et al., 2006) which uses a top-N list of translations to select the maximum BLEU sentence as a target for training (so called local update).

3 A Categorization of Block Styles

In (Brown et al., 1993), multi-word "cepts" (which are realized in our block concept) are discussed and the authors state that when a target sequence is sufficiently different from a word by word translation, only then should the target sequence should be promoted to a cept. This is in direct opposition to phrase-based decoders which utilize all possible phrase-pairs and limit the number of phrases only due to practical considerations. Following the perspective of (Brown et al., 1993), a minimal set of phrase blocks with lengths (m, n) where either m or n must be greater than zero results in the following types of blocks:

1. $n = 0$, source word producing nothing in the target language (deletion block),
2. $m = 0$, spontaneous target word (insertion block),
3. $m = 1$ and $n \geq 1$, a source word producing n target words including the possibility of a variable (denoted by $\mathbf{X}$) which is to be filled with other blocks from the sentence (the latter case called a discontiguous block)
4. $m \geq 1$ and $n = 1$, a sequence of source words producing a single target words including the possibility of a variable on the source side (as in the French ne...pas translating into not, called multi-word singletons) in the source sequence

5. $m > 1$ and $n > 1$, a non-compositional phrase translation

In this paper, we restrict the blocks to Types 1 and 3. From the example in Figure 1, the following blocks are extracted:

- lljnp ⇒ of the **X** Committee
- Almrkzyp ⇒ Central
- llHzb ⇒ of the **X** Party
- Al$ywEy ⇒ Communist
- AlSyny ⇒ Chinese.

These blocks can now be considered more "general" and can be used to generate more phrases compared to the blocks shown in Table 1. These blocks when utilized independently of the remainder of the model perform very poorly as all the advantages of blocks are absent. These advantages are obtained using the features to be described below. Also, we store with a block additional information such as: (a) alignment information, and (b) source and target analysis. The target analysis includes part of speech and for each target string a list of part of speech sequences are stored along with their corpus frequencies.

The first alignment shown in Figure 1 is an example of a Type 5 non-compositional block; although this is not currently addressed by the decoder, we plan to handle such blocks in the future.

4 Algorithm

A classification problem can be considered as a mapping from a set of histories, $\mathcal{S}$, into a set of futures, $\mathcal{T}$. Traditional classification problems deal with a small finite set of futures usually no more than a few thousands of classes.

Machine translation can be cast into the same framework with a much larger future space. In contrast to the current global models, we decompose the process into a sequence of steps. The process begins at the left edge of a sentence and for practical reasons considers a window of source words that could be translated. The first action is to jump a distance, j to a source position and to produce a target string, $\boldsymbol{t}$ corresponding to the source word at that position. The process then marks the source position as having been visited and iterates till all source words have been visited. The only wrinkle in this relatively simple process is the presence of a variable in the target sequence. In the case of a variable, the source position is marked as having been partially visited. When a partially visited source position is visited again, the target string to the right of the variable is output and the process is iterated. The distortion or jump from the previously translated source word, j in training can vary widely due to automatic sentence alignment that is used to create the parallel corpus. To limit the sparseness created by these longer jumps we cap the jump to a window of source words (-5 to 5 words) around the last translated source word; jumps outside the window are treated as being to the edge of the window.

We combine the above translation model with a n-gram language model as in

$$\begin{aligned} p(T, j|S) &= \prod_i p(t_i, j|s_i) \\ &\approx \prod_i \lambda_{\text{LM}} p(t_i|t_{i-1}, \ldots, t_{i-n}) + \\ &\qquad \lambda_{\text{TM}} p(t_i, j|s_i) \end{aligned}$$

This mixing allows the use of language model built from a very large monolingual corpus to be used with a translation model which is built from a smaller parallel corpus. In the rest of this paper, we are concerned only with the translation model.

The minimum requirements for the algorithm are (a) parallel corpus of source and target languages and (b) word-alignments. While one can use the EM algorithm to train this hidden alignment model (the jump step), we use Viterbi training, i.e. we use the most likely alignment between target and source words in the training corpus to estimate this model. We assume that each sentence pair in the training corpus is word-aligned (e.g. using a MaxEnt aligner (Ittycheriah and Roukos, 2005) or an HMM aligner (Ge, 2004)). The algorithm performs the following steps in order to train the maximum entropy model: (a) block extraction, (b) feature extraction, and (c) parameter estimation. Each of the first two steps requires a pass over the training data and parameter estimation requires typically 5-10 passes over the data. (Della Pietra et al., 1995) documents the Improved Iterative Scaling (IIS) algorithm for training maximum entropy models. When the system is restricted to 1-N type blocks, the future space includes all the source word positions that are within the skip window and all their corresponding blocks. The training algorithm at the parameter estimation step can be concisely stated as:

1. For each sentence pair in the parallel corpus, walk the alignment in source word order.
2. At each source word, the alignment identifies the "true" block.
3. Form a window of source words and allow all blocks at source words to generate at this generation point.

4. Apply the features relevant to each block and compute the probability of each block.

5. Form the MaxEnt polynomials(Della Pietra et al., 1995) and solve to find the update for each feature.

We will next discuss the prior distribution used in the maximum entropy model, the block extraction method and the feature generation method and discuss differences with a standard phrase based decoder.

4.1 Prior Distribution

Maximum entropy models are of the form,

$$p(\boldsymbol{t}, j|\boldsymbol{s}) = \frac{p_0(\boldsymbol{t}, j|\boldsymbol{s})}{Z} \exp \sum_i \lambda_i \phi_i(\boldsymbol{t}, j, \boldsymbol{s})$$

where p_0 is a prior distribution, Z is a normalizing term, and $\phi_i(\boldsymbol{t}, j, \boldsymbol{s})$ are the features of the model. The prior distribution can contain any information we know about our future and in this work we utilize the normalized phrase count as our prior. Strictly, the prior has to be uniform on the set of futures to be a "maximum" entropy algorithm and choices of other priors result in minimum divergence models. We refer to both as a maximum entropy models.

The practical benefit of using normalized phrase count as the prior distribution is for rare translations of a common source words. Such a translation block may not have a feature due to restrictions in the number of features in the model. Utilizing the normalized phrase count prior, the model is still able to penalize such translations. In the best case, a feature is present in the model and the model has the freedom to either boost the translation probability or to further reduce the prior.

4.2 Block Extraction

Similar to phrase decoders, a single pass is made through the parallel corpus and for each source word, the target sequence derived from the alignments is extracted. The 'Inverse Projection Constraint', which requires that the target sequence be aligned only to the source word or phrase in question, is then checked to ensure that the phrase pair is consistent. A slight relaxation is made to the traditional target sequence in that variables are allowed if the length of their span is 3 words or less. The length restriction is imposed to reduce the effect of alignment errors. An example of blocks extracted for the romanized arabic words 'lljnp' and 'Almrkzyp' are shown Figure 2, where on the left side are shown the unsegmented Arabic words, the segmented Arabic stream and the corresponding Arabic part-of-speech. On the right, the target sequences are shown with the most frequently occuring part-of-speech and the corpus count of this block.

The extracted blocks are pruned in order to minimize alignment problems as well as optimize the speed during decoding. Blocks are pruned if their corpus count is a factor of 30 times smaller than the most frequent target sequence for the same source word. This results in about 1.6 million blocks from an original size of 3.2 million blocks (note this is much smaller than the 50 million blocks or so that are derived in current phrase-based systems).

4.3 Features

The features investigated in this work are binary questions about the lexical context both in the source and target streams. These features can be classified into the following categories: (a) block internal features, and (b) block context features. Features can be designed that are specific to a block. Such features are modeling the unigram phrase count of the block, which is information already present in the prior distribution as discussed above. Features which are less specific are tied across many translations of the word. For example in Figure 2, the primary translation for 'lljnp' is 'committee' and occurs 920 times across all blocks extracted from the corpus; the final block shown which is 'of the **X** committee' occurs only 37 times but employs a lexical feature 'lljnp committee' which fires 920 times.

4.3.1 Lexical Features

Lexical features are block internal features which examine a source word, a target word and the jump from the previously translated source word. As discussed above, these are shared across blocks.

4.3.2 Lexical Context Features

Context features encode the context surrounding a block by examining the previous and next source word and the previous two target words. Unlike a traditional phrase pair, which encodes all the information lexically, in this approach we define in Table 2, individual feature types to examine a portion of the context. One or more of these features may apply in each instance where a block is relevant. The previous source word is defined as the previously translated source word, but the next source word is always the next word in the source string. At training time, the previously translated source word is found by finding the previous target word and utilizing the alignment to find the previous source word. If the previous target word is unaligned, no context feature is applied.

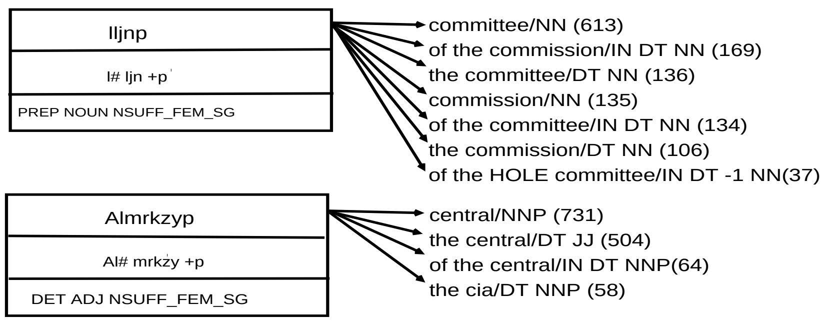

lljnp
l# ljn +p
PREP NOUN NSUFF_FEM_SG
committee/NN (613)
of the commission/IN DT NN (169)
the committee/DT NN (136)
commission/NN (135)
of the committee/IN DT NN (134)
the commission/DT NN (106)
of the HOLE committee/IN DT -1 NN(37)
Almrkzyp
Al# mrkzy +p
DET ADJ NSUFF_FEM_SG
central/NNP (731)
the central/DT JJ (504)
of the central/IN DT NNP(64)
the cia/DT NNP (58)

Figure 2: Extracted blocks for 'lljnp' and 'Almrkzyp'.

Feature Name	Feature variables
SRC_LEFT	source left, source word, target word
SRC_RIGHT	source right, source word, target word
SRC_TGT_LEFT	source left, target left, source word, target word
SRC_TGT_LEFT_2	source left, target left, target left 2, source word, target word

Table 2: Context Feature Types

4.3.3 Arabic Segmentation Features

An Arabic segmenter produces morphemes; in Arabic, prefixes and suffixes are used as prepositions, pronouns, gender and case markers. This produces a segmentation view of the arabic source words (Lee et al., 2003). The features used in the model are formed from the Cartesian product of all segmentation tokens with the English target sequence produced by this source word or words. However, prefixes and suffixes which are specific in translation are limited to their English translations. For example the prefix 'Al#' is only allowed to participate in a feature with the English word 'the' and similarly 'the' is not allowed to participate in a feature with the stem of the Arabic word. These restrictions limit the number of features and also reduce the over fitting by the model.

4.3.4 Part-of-speech Features

Part-of-speech taggers were run on each language: the English part of speech tagger is a MaxEnt tagger built on the WSJ corpus and on the WSJ test set achieves an accuracy of 96.8%; the Arabic part of speech tagger is a similar tagger built on the Arabic tree bank and achieves an accuracy of 95.7% on automatically segmented data. The part of speech feature type examines the source and target as well as the previous target and the corresponding previous source part of speech. A separate feature type examines the part of speech of the next source word when the target sequence has a variable.

4.3.5 Coverage Features

These features examine the coverage status of the source word to the left and the source word to the right. During training, the coverage is determined by examining the alignments; the source word to the left is uncovered if its target sequence is to the right of the current target sequence. Since the model employs binary questions and predominantly the source word to the left is already covered and the right source word is uncovered, these features fire only if the left is open or if the right is closed in order to minimize the number of features in the model.

5 Translation Decoder

A beam search decoder similar to phrase-based systems (Tillmann and Ney, 2003) is used to translate the Arabic sentence into English. These decoders have two parameters that control their search strategy: (a) the skip length (how many positions are allowed to be untranslated) and (b) the window width, which controls how many words are allowed to be considered for translation. Since the majority of the blocks employed in this work do not encode local reordering explicitly, the current DTM2 decoder uses a large skip (4 source words for Arabic) and tries all possible reorderings. The primary difference between a DTM2 decoder and standard phrase based decoders is that the maximum entropy model provides a cost estimate of producing this translation using the features described in previous sections. Another difference is that the DTM2 decoder handles blocks with variables. When such a block is proposed, the initial target sequence is first output and the source word position is marked as being partially visited and an index into which segment was generated is kept for completing the visit at a later time. Subsequent extensions of this path can either complete this visit or visit other source words. On a search path, we make a further assumption that only one source position can be in a partially visited state at any point. This greatly reduces the search task and suffices to handle the type of blocks encountered in Arabic to English translation.

6 Experiments

The UN parallel corpus and the LDC news corpora released as training data for the NIST MT06 evaluation are used for all evaluations presented in this paper. A variety of test corpora are now available and we use MT03 as development test data, and test results are presented on MT05. Results obtained on MT06 are from a blind evaluation. For Arabic-English, the NIST MT06 training data contains 3.7M sentence pairs from the UN from 1993-2002 and 100K sentences pairs from news sources. This represents the universe of training data, but for each test set we sample this corpus to train efficiently while also observing slight gains in performance. The training universe is time sorted and the most recent corpora are sampled first. Then for a given test set, we obtain the first 20 instances of n-grams from the test that occur in the training universe and the resulting sampled sentences then form the training sample. The contribution of the sampling technique is to produce a smaller training corpus which reduces the computational load; however, the sampling of the universe of sentences can be viewed as test set domain adaptation which improves performance and is not strictly done due to computational limitations[2]. The 5-gram language model is trained from the English Gigaword corpus and the English portion of the parallel corpus used in the translation model training.

The baseline decoder is a phrase-based decoder that employs *n-m* blocks and uses the same test set specific training corpus described above.

6.1 Feature Type Experiments

There are 15 individual feature types utilized in the system, but in order to be brief we present the results by feature groups (see Table 3): (a) lexical, (b) lexical context, (c) segmentation, (d) part-of-speech, and (e) coverage features. The results show improvements with the addition of each feature set, but the part-of-speech features and coverage features are not statistically significant improvements. The more complex features based on Arabic segmentation and English part-of-speech yield a small improvement of 0.5 BLEU points over the model with only lexical context.

[2]Recent results indicate that test set adaptation by test set sampling of the training corpus achieves a cased Bleu of 53.26 on MT03 whereas a general system trained on all data achieves only 51.02

Verb Placement	3
Missing Word	5
Extra Word	5
Word Choice	26
Word Order	3
Other error	1
Total	43

Table 4: Errors on last 25 sentences of MT-03.

7 Error Analysis and Discussion

We analyzed the errors in the last 25 sentences of the MT-03 development data using the broad categories shown in Table 4. These error types are not independent of each other; indeed, incorrect verb placement is just a special case of the word order error type but for this error analysis for each error we take the first category available in this list. Word choice errors can be a result of (a) rare words with few, or incorrect, or no translation blocks (4 times) or (b) model weakness[3] (22 times). In order to address the model weakness type of errors, we plan on investigating feature selection using a language model prior. As an example, consider an arabic word which produces both 'the' (due to alignment errors) and 'the conduct'. An n-gram LM has very low cost for the word 'the' but a rather high cost for content words such as 'conduct'. Incorporating the LM model as a prior should help the maximum entropy model focus its weighting on the content word to overcome the prior information.

8 Conclusion and Future Work

We have presented a complete direct translation model with training of millions of parameters based on a set of minimalist blocks and demonstrated the ability to retain good performance relative to phrase based decoders. Tied features minimize the number of parameters and help avoid the sparsity problems associated with phrase based decoders. Utilizing language analysis of both the source and target languages adds 0.8 BLEU points on MT-03, and 0.4 BLEU points on MT-05. The DTM2 decoder achieved a 1.7 BLEU point improvement over the phrase based decoder on MT-06. In this work, we have restricted the block types to only single source word blocks. Many city names and dates in Arabic can not be handled by such blocks and in future work we intend to investigate the utilization of more complex blocks as necessary. Also, the DTM2 decoder utilized the LM component independently of

[3]The word occurred with the correct translation in the phrase library with a count more than 10 and yet the system used an incorrect translation.

Feature Types		# of feats (MT03)	MT-03	MT-05	MT-06
Training Size Num. of Sentences			197K	267K	279K
Phrase-based Decoder			51.20	49.06	36.92
DTM2 Decoder Lex Feats	a	439,582	49.70	48.37	
+Lex Context	b	2,455,394	50.45	49.61	
+Seg Feats	c	2,563,338	50.97	49.96	
+POS Feats	d	2,608,352	51.27	49.93	
+Cov Feats	e	2,783,813	51.19	50.00	38.61

Table 3: Bleu scores on MT03-MT06.

the translation model; however, in future work we intend to investigate feature selection using the language model as a prior which should result in much smaller systems.

9 Acknowledgements

This work was partially supported by the Department of the Interior, National Business Center under contract No. NBCH2030001 and Defense Advanced Research Projects Agency under contract No. HR0011-06-2-0001. 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

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 529–536, Sydney, Australia.

Rafael Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, and José B. Marino. 2005. Statistical machine translation of euparl data by using bilingual n-grams. In *Proc. of the ACL Workshop on Building and Using Parallel Texts*, pages 133–136, Ann Arbor, Michigan, USA.

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.

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, June.

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*.

George Foster. 2000. A maximum entropy/minimum divergence translation model. In *38th Annual Meeting of the ACL*, pages 45–52, Hong Kong.

Niyu Ge. 2004. Improvement in Word Alignments. *Presentation given at DARPA/TIDES MT workshop.*

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *HLT '05: Proceedings of the HLT and EMNLP*, pages 89–96.

Young-Suk Lee, Kishore Papineni, and Salim Roukos. 2003. Language model based arabic word segmentation. In *41st Annual Meeting of the ACL*, pages 399–406, Sapporo, Japan.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 761–768, Sydney, Australia.

Franz Josef Och and Hermann Ney. 2000. Statistical machine translation. In *EAMT Workshop*, pages 39–46, Ljubljana, Slovenia.

Franz-Josef Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translations. In *40th Annual Meeting of the ACL*, pages 295–302, Philadelphia, PA, July.

Franz Josef Och. 2003. Minimum error rate training in Statistical Machine Translation. In *41st Annual Meeting of the ACL*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, and R. T. Ward. 1997. Feature-based language understanding. In *EUROSPEECH*, pages 1435–1438, Rhodes,Greece.

Kishore Papineni, Salim Roukos, and R. T. Ward. 1998. Maximum likelihood and discriminative training of direct translation models. In *International Conf. on Acoustics, Speech and Signal Processing*, pages 189–192, Seattle, WA.

Chris Quirk and Arul Menezes. 2006. Do we need phrases? challenging the conventional wisdom in statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 9–16, New York, NY, USA.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for Statistical Machine Translation. 29(1):97–133.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical mt. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 721–728, Sydney, Australia.

Structured Local Training and Biased Potential Functions for Conditional Random Fields with Application to Coreference Resolution

Yejin Choi and Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY 14853
{ychoi,cardie}@cs.cornell.edu

Abstract

Conditional Random Fields (CRFs) have shown great success for problems involving structured output variables. However, for many real-world NLP applications, exact maximum-likelihood training is intractable because computing the global normalization factor even approximately can be extremely hard. In addition, optimizing likelihood often does not correlate with maximizing task-specific evaluation measures. In this paper, we present a novel training procedure, *structured local training*, that maximizes likelihood while exploiting the benefits of global inference during training: hidden variables are used to capture interactions between local inference and global inference. Furthermore, we introduce *biased potential functions* that empirically drive CRFs towards performance improvements w.r.t. the preferred evaluation measure for the learning task. We report promising experimental results on two coreference data sets using two task-specific evaluation measures.

1 Introduction

Undirected graphical models such as Conditional Random Fields (CRFs) (Lafferty et al., 2001) have shown great success for problems involving structured output variables (e.g. Wellner et al. (2004), Finkel et al. (2005)). For many real-world NLP applications, however, the required graph structure can be very complex, and computing the global normalization factor even approximately can be extremely hard. Previous approaches for training CRFs have either (1) opted for a training method that no longer maximizes the likelihood, (e.g. McCallum and Wellner (2004), Roth and Yih (2005)) [1], or (2) opted for a simplified graph structure to avoid intractable global normalization (e.g. Roth and Yih (2005), Wellner et al. (2004)).

Solutions of the first type replace the computation of the global normalization factor $\sum_{\mathbf{y}} \; p(\mathbf{y}|\mathbf{x})$ with $\text{argmax}_{\mathbf{y}} \; p(\mathbf{y}|\mathbf{x})$ during training, since finding an argmax of a probability distribution is often an easier problem than finding the entire probability distribution. Training via the voted perceptron algorithm (Collins, 2002) or using a max-margin criterion also correspond to the first option (e.g. McCallum and Wellner (2004), Finley and Joachims (2005)). But without the global normalization, the maximum-likelihood criterion motivated by the maximum entropy principle (Berger et al., 1996) is no longer a feasible option as an optimization criterion.

The second solution simplifies the graph structure for training, and applies complex global inference only for testing. In spite of the discrepancy between the training model and the testing model, it has been empirically shown that (1) performing global inference only during testing can improve performance (e.g. Finkel et al. (2005), Roth and Yih (2005)), and (2) full-blown global training can often perform worse due to insufficient training data (e.g. Punyakanok et al. (2005)). Importantly, however, attempts to reduce the discrepancy between the training and test models — by judiciously adding the effect of global inference to the training — have produced substantial performance improvements over locally trained models (e.g. Cohen and Carvalho (2005), Sutton and McCallum (2005a)).

In this paper, we present *structured local training*, a novel training procedure for maximum-likelihood

[1] Both McCallum and Wellner (2004) and Roth and Yih (2005) used the voted perceptron algorithm (Collins, 2002) to train intractable CRFs.

training of undirected graphical models, such as CRFs. The procedure maximizes likelihood while exploiting the benefits of global inference during training by capturing the interactions between local inference and global inference via hidden variables.

Furthermore, we introduce *biased potential functions* that redefine the likelihood for CRFs so that the performance of CRFs trained under the maximum likelihood criterion correlates better empirically with the preferred evaluation measures such as F-score and MUC-score.

We focus on the problem of coreference resolution; however, our approaches are general and can be extended to other NLP applications with structured output. Our approaches also extend to non-conditional graphical models such as Markov Random Fields. In experiments on two coreference data sets, structured local training reduces the error rate significantly (3.5%) for one coreference data set and minimally ($\leq$ 1%) for the other. Experiments using biased potential functions increase recall uniformly and significantly for both data sets and both task-specific evaluation measures. Results for the combination of the two techniques are promising, but mixed: pairwise F1 increases by 0.8-5.5% for both data sets; MUC F1 increases by 3.5% for one data set, but slightly hurts performance for the second data set.

In §2, we describe *structured local training*, and follow with experimental results in §3. In §4, we describe *biased potential functions* and follow with experimental results in §5. We discuss related work in §6.

2 Structured Local Training

2.1 Definitions

For clarity, we define the following terms that we will use throughout the paper.

- **local inference:** [2] Inference factored into smaller independent pieces, without considering the structure of the output space.
- **global inference:** Inference applied on the entire set of output variables, considering the structure of the output space.
- **local training:** Training that does not invoke global inference at each iteration.
- **global training:** Training that does invoke global inference at each iteration.

[2]In this paper, inference refers to the operation of finding the *argmax* in particular.

2.2 A Motivating Example for Coreference Resolution

In this section, we present an example of the coreference resolution problem to motivate our approach. It has been shown that global inference-based training for coreference resolution outperforms training with local inference only (e.g. Finley and Joachims (2005), McCallum and Wellner (2004)). In particular, the output of coreference resolution must obey equivalence relations, and exploiting such structural constraints on the output space during training can improve performance. Consider the coreference resolution task for the following text.

> It was after the passage of this act, that $\underline{\text{Mary}^{(1)}}$'s attitude towards $\underline{\text{Elizabeth}^{(1)}}$ became overtly hostile. The deliberations surrounding the act seem to have revived all Mary's memories of the humiliations she had suffered at the hands of Anne Boleyn. At the same time, $\underline{\text{Elizabeth}^{(2)}}$'s continuing prevarications over religion confirmed that **she** was indeed her mother's daughter.

In the above text, the "she" in the last sentence is coreferent with both mentions of "$Elizabeth$". However, when we consider "she" and "$Elizabeth^{(1)}$" in isolation from the remaining coreference chain, it can be difficult for a machine learning method to determine whether the pair is coreferent or not. Indeed, such a pair may not look very different from the pair "she" and "$Mary^{(1)}$" in terms of feature vectors. It is much easier, however, to determine that "she" and "$Elizabeth^{(2)}$" are coreferent, or that "$Elizabeth^{(1)}$" and "$Elizabeth^{(2)}$" are coreferent. Only by taking the transitive closure of these pairwise coreference relations does it become clear that "she" and "$Elizabeth^{(1)}$" are coreferent. In other words, global training might handle potentially confusing coreference cases better because it allows parameter learning (for each pairwise coreference decision) to be informed by global inference.

We argue that, with appropriate modification to the learning instances, local training is adequate for the coreference resolution task. Specifically, we propose that confusing pairs in the training data — such as "she" and "$Elizabeth^{(1)}$" — be learned as *not-coreferent*, so long as the global inference step can fix this error by exploiting the structure of the output space, i.e. by exploiting the equivalence relations. This is the key idea of *structured local training*, which we elaborate formally in the following section.

2.3 A Hidden-Variable Model

In this section, we present a general description of *structured local training*. Let **y** be a vector of output variables for structured output, and let **x** be a vector of input variables. In order to capture the interactions between global inference and local inference, we introduce hidden variables **h**, $|\mathbf{h}| = |\mathbf{y}|$, so that the global inference for $p(\mathbf{y}, \mathbf{h}|\mathbf{x})$ can be factored into two components using the product rule, as follows:

$$\begin{aligned} p(\mathbf{y}, \mathbf{h}|\mathbf{x}) &= p(\mathbf{y}|\mathbf{h}, \mathbf{x})\, p(\mathbf{h}|\mathbf{x}) \\ &= p(\mathbf{y}|\mathbf{h})\, p(\mathbf{h}|\mathbf{x}) \end{aligned}$$

The second component $p(\mathbf{h}|\mathbf{x})$ on the right hand side corresponds to the local model, for which the inference factorizes into smaller independent pieces, e.g. $\text{argmax}_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}) = \{\text{argmax}_{h_i} \phi(h_i, \mathbf{x})\}$. And the first component $p(\mathbf{y}|\mathbf{h}, \mathbf{x})$ on the right hand side corresponds to the global model, whose inference may not factorize nicely. Further, we assume that **y** is independent of **x** given **h**, so that $p(\mathbf{y}|\mathbf{h}, \mathbf{x}) = p(\mathbf{y}|\mathbf{h})$. That is to say, **h** captures sufficient information from **x**, so that given **h**, global inference of **y** only depends on **h**. The quantity of $p(\mathbf{y}|\mathbf{x})$ then is given by marginalizing out **h** as follows:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h}|\mathbf{x})$$

Intuitively, the hidden variables **h** represent the local decisions that can lead to a good **y** after global inference is applied. In the case of coreference resolution, one natural factorization would be that global inference is a clustering algorithm, and local inference is a classification decision on each pair of noun phrases (or mentions).[3] In this paper, we assume that we only parameterize the local model $p(\mathbf{h}|\mathbf{x})$, although it would be possible to extend the parameterization to the global model as well, depending on the particular application under consideration. The similarity between a pair of mentions is parameterized via log-linear models. However, once we have the similarity scores extracted via local inference, the clustering algorithm does not require further parameterization.

For training, we apply the standard Expectation-Maximization (EM) algorithm (Dempster et al., 1977) as follows:

- E Step: Compute a distribution

$$\widetilde{P}^{(t)} = P(\mathbf{h}|\mathbf{y}, \mathbf{x}, \theta^{(t-1)})$$

- M Step: Set $\theta^{(t)}$ to θ that maximizes

$$E_{\widetilde{P}^{(t)}} [\log P(\mathbf{y}, \mathbf{h}|\mathbf{x}, \theta)]$$

By repeatedly applying the above two steps for $t = 1, 2, ...$, the value of θ converges to the local maxima of the conditional log likelihood $L(\theta) = \log P(\mathbf{y}|\mathbf{x}, \theta)$.

2.4 Application to Coreference Resolution

For $y_i \in \mathbf{y}$ (and $h_i \in \mathbf{h}$) in the coreference resolution task, $y_i = 1$ (and $h_i = 1$) corresponds to ith pair of mentions being coreferent, and $y_i = 0$ (and $h_i = 0$) corresponds to ith pair being not coreferent.

[Local Model $P(\mathbf{h}|\mathbf{x})$] For the local model, we define cliques as individual nodes,[4] and parameterize each clique potential as

$$\phi(h_i, \mathbf{x}) = \phi(h_i, x_i) = \exp \sum_k \lambda_k f_k(h_i, x_i)$$

Let $\Phi(\mathbf{h}|\mathbf{x}) \equiv \prod_i \phi(h_i, x_i)$. Then,

$$P(\mathbf{h}|\mathbf{x}) = \frac{\Phi(\mathbf{h}, \mathbf{x})}{\sum_{\mathbf{h}} \Phi(\mathbf{h}, \mathbf{x})}$$

Notice that in this model, finding $\text{argmax}_{\mathbf{h}} P(\mathbf{h}|\mathbf{x})$ corresponds to simply finding $\text{argmax}_{h_i} \phi(h_i, x_i)$ independently for each $h_i \in \mathbf{h}$.

[3]Formally, we define each $y_i \in \mathbf{y}$ to be the coreference decision for the ith pair of mentions, and $x_i \in \mathbf{x}$ be the input regarding the ith pair of mentions. Then h_i corresponds to the local coreference decision that can lead to a good coreference decision y_i after the clustering algorithm has been applied.

[4]Each node in the graphical representation of CRFs corresponds to the coreferent decision for each pair of mentions. This corresponds to the "Model 3" of McCallum and Wellner (2004).

```
ALGORITHM-1
INPUT: $\mathbf{x}$, true labeling $\mathbf{y}^*$, current local model $P(\mathbf{h}|\mathbf{x})$
GOAL: Find the highest confidence labeling $\mathbf{y}'$
      such that $\mathbf{y}^* =$ single-link-clustering($\mathbf{y}'$)
$\mathbf{h}^* \leftarrow \text{argmax}_{\mathbf{h}} P(\mathbf{h}|\mathbf{x})$
$\mathbf{h}' \leftarrow$ single-link-clustering($\mathbf{h}^*$)
construct a graph $G = (V, E)$, where
    $E = \{h'_i : h'_i \in \mathbf{h}' \text{ s.t. } y^*_i = 1\}$
    $V = \{v : v$ is a NP referred by a $h'_i \in E\}$
    with edge cost $cost_{h'_i} = \phi(h'_i, x_i)$ if $h'_i \neq y^*_i$
    with edge cost $cost_{h'_i} = 0$ if $h'_i = y^*_i$
find a minimum spanning tree(or forest) $M$ of $G$
for each $h'_i \in \mathbf{h}'$
    if $h'_i = y^*_i$
        $y'_i \leftarrow h^*_i$
    else if $h'_i \in M$
        $y'_i \leftarrow 1$
    else
        $y'_i \leftarrow 0$
end for
return $\mathbf{y}'$
```

Figure 1: Algorithm to find the highest confidence labeling $\mathbf{y}'$ that can be clustered to the true labeling $\mathbf{y}^*$

```
ALGORITHM-2
INPUT: $\mathbf{x}$, true labeling $\mathbf{y}^*$, current local model $P(\mathbf{h}|\mathbf{x})$
GOAL: Find a high confidence labeling $\mathbf{y}'$ that is
      close to the true labeling $\mathbf{y}^*$
$\mathbf{h}^* \leftarrow \text{argmax}_{\mathbf{h}} P(\mathbf{h}|\mathbf{x})$
$\mathbf{h}' \leftarrow$ single-link-clustering($\mathbf{h}^*$)
for each $h'_i \in \mathbf{h}'$
    if $h'_i = y^*_i$
        $y'_i \leftarrow h^*_i$
    else
        $y'_i \leftarrow y^*_i$
end for
return $\mathbf{y}'$
```

Figure 2: Algorithm to find a high confidence labeling $\mathbf{y}'$ that is close to the true labeling $\mathbf{y}^*$

[Global Model $P(\mathbf{y}|\mathbf{h})$] For the global model, we assume a deterministic clustering algorithm is given. In particular, we focus on single-link clustering, as it has been shown to be effective for coreference resolution (e.g. Ng and Cardie (2002)). With single-link clustering, $P(\mathbf{y}|\mathbf{h}) = 1$ if $\mathbf{h}$ can be clustered to $\mathbf{y}$, and $P(\mathbf{y}|\mathbf{h}) = 0$ if $\mathbf{h}$ cannot be clustered to $\mathbf{y}$.[5]

[Computation of the E-step] The E-step requires computation of the distribution of $P(\mathbf{h}|\mathbf{y}, \mathbf{x}, \theta^{(t-1)})$, which we will simply denote as $P(\mathbf{h}|\mathbf{y}, \mathbf{x})$, since all our distributions are implicitly conditioned on the model parameters θ.

$$P(\mathbf{h}|\mathbf{y}, \mathbf{x}) = \frac{P(\mathbf{h}, \mathbf{y}|\mathbf{x})}{P(\mathbf{y}|\mathbf{x})} \propto P(\mathbf{y}|\mathbf{h})\ P(\mathbf{h}|\mathbf{x})$$

Notice that when computing $P(\mathbf{h}|\mathbf{y}, \mathbf{x})$, the denominator $P(\mathbf{y}|\mathbf{x})$ stays as a constant for different values of $\mathbf{h}$. The E-step requires enumeration of all possible values of $\mathbf{h}$, but it is intractable with our formulation, because inference for the global model $P(\mathbf{y}|\mathbf{h})$ does not factor out nicely. Therefore, we must resort to an approximation method. Neal and Hinton (1998) analyze and motivate various approximate EM training methods. One popular choice in practice is called "Viterbi training", a variant of the EM algorithm, which has been shown effective in many NLP applications. Viterbi training approximates the distribution by assigning all probability mass to a single best assignment. The algorithm for this is shown in Figure 1.

We propose another approximation option for the E-step that is given by Figure 2. Intuitively, when the current local model misses positive coreference decisions, the first algorithm constructs a $\mathbf{y}'$ that is closest to $\mathbf{h}'$ for single-link clustering to recover the true labeling $\mathbf{y}^*$, while the second algorithm constructs a $\mathbf{y}'$ that is closer to $\mathbf{y}^*$ by preserving all of the missing positive coreference decisions. [6]

[Computation of M-step] Because $P(\mathbf{y}|\mathbf{h})$ is not parameterized, finding argmax$_\theta$ $P(\mathbf{y}, \mathbf{h}|\mathbf{x})$ reduces to finding argmax$_\theta$ $P(\mathbf{h}|\mathbf{x})$, which is standard CRF training. In order to speed up the training, we start convex optimization for CRFs using the parameter values $\theta^{(t-1)}$ from the previous M-step. For the very first iteration of EM, we start by setting $P(\mathbf{y}^*|\mathbf{x}) = 1$ for E-step, so that the first M-step will finds argmax$_\theta$ $P(\mathbf{y}^*|\mathbf{x})$.

[5]Single-link clustering simply takes the transitive closure, and does not consider the distance metric. In a pilot study, we also tried a variant of a stochastic clustering algorithm that takes into account the distance metric (set as the probabilities from the local model) for the global model, but the performance was worse.

[6]In a pilot study, we found that ALGORITHM-2 performs slightly better than ALGORITHM-1. We also tried two other approximation options, but none performed as well as ALGORITHM-2. One of them removes the confusing sub-instances and has the effect of setting a uniform distribution on those sub-instances. The other computes the actual distribution on a subset of sub-instances. For brevity, we only present experimental results using ALGORITHM-2 in this paper.

[Inference on the test data] It is intractable to marginalize out **h** from $P(\mathbf{y}, \mathbf{h}|\mathbf{x})$. Therefore, similar to the Viterbi-training in the E-step, we approximate the distribution of **h** by $\text{argmax}_{\mathbf{h}} P(\mathbf{h}|\mathbf{X})$.

3 Experiments–I

Data set: We evaluate our approach with two coreference data sets: MUC6 (MUC-6, 1995) and MPQA[7](Wiebe et al., 2005). For the MUC6 data set, we extract noun phrases (mentions) automatically, but for MPQA, we assume mentions for coreference resolution are given as in Stoyanov and Cardie (2006). For MUC6, we use the standard training/test data split. For MPQA, we use 150 documents for training, and 50 documents for testing.

Configuration: We follow Ng and Cardie (2002) for feature vector construction for each pair of mentions,[8] and Finley and Joachims (2005) for constructing a training/testing instance for each document: a training/testing instance consists of all pairs of mentions in a document. Then, a single pair of mentions is a *sub-instance*. We use the Mallet[9] implementation of CRFs, and set a Gaussian prior of 1.0 for all experiments. At each M-step, we train CRFs starting from the parameters from the previous M-step. We train CRFs up to 200 iterations, but because we start training CRFs from the previous parameters, the convergence from the second M-step becomes much faster. We apply up to 5 EM iterations, and choose best performing $\theta^{(t)}, 2 \leq t \leq 5$ based on the performance on the training data.[10]

Hypothesis: For the baseline (BASE) we employ the locally trained model for pairwise decisions without global inference. Clustering is applied only at test time, in order to make the assignment on the output variables coherent. We hypothesize that for the baseline, maximizing the likelihood for training will correlate more with the pairwise accuracy of the incoherent decisions before clustering than the pairwise accuracy of the coherent decisions after clustering. We also hypothesize that by performing structured local training (SLT), maximizing the likelihood will correlate more with the pairwise accuracy after clustering.

MUC6								
	after clustering				before clustering			
	e %	R %	P %	F %	e %	R %	P %	F %
BASE	1.50	**59.2**	56.2	**57.7**	1.18	38.0	85.6	52.6
SLT	**1.28**	49.8	**67.3**	57.2	1.35	26.4	84.3	40.2
MPQA								
	after clustering				before clustering			
	e %	R %	P %	F %	e %	R %	P %	F %
BASE	9.83	**75.8**	57.0	65.1	7.05	52.1	83.4	64.1
SLT	**6.39**	62.1	**80.6**	**70.2**	7.39	43.7	90.1	58.9

Table 1: Performance of Structured Local Training: SLT reduces error rate (e %) after applying single-link clustering.

Results: Experimental results are shown in Table 1. We report error rate $(\text{error rate} = 100 - \text{accuracy})$ on the pairwise decisions (e %), and F1-score (F %) on the coreferent pairs.[11] For comparison, we show numbers from both after and before single-link clustering is applied. As hypothesized, the error rate of BASE increases after clustering, while the error rate of SLT decreases after clustering. Moreover, the error rate of SLT is considerably lower than that of BASE after clustering. However, the F1-score does not correlate with the error rate. That is, a lower error rate does not always lead to a higher F1-score, which motivates the *Biased Potential Functions* that we introduce in the next section. Notice that when we compare the precision/recall breakdown after clustering, SLT has higher precision and lower recall than BASE.

4 Biased Potential Functions

We introduce *biased potential functions* for training CRFs to empirically favor preferred evaluation measures for the learning task, such as F-score and MUC-score that have been considered hard for tradi-

[7]Available at http://nrrc.mitre.org/NRRC/publications.htm.

[8]In particular, our feature set corresponds to "All Features" in Ng and Cardie (2002), and we discretized numeric values.

[9]Available at http://mallet.cs.umass.edu.

[10]Selecting $\theta^{(t)}$ on a separate tuning data would be better, but the data for MUC6 in particular is very limited. Notice that we don't pick θ^1 when reporting the performance of SLT, because it is identical to the baseline.

[11]Error rate and F1-score on the coreferent pairs are not ideal measures for the quality of clustering, however, we show them here in order to contrast the effect of SLT. We present MUC-scores for the same experimental settings in Table 3.

tional likelihood-based methods to optimize for. Intuitively, biased potential functions emphasize those sub-components of an instance that can be of greater importance than the rest of an instance.

4.1 Definitions

The conditional probability of $P(\mathbf{y}|\mathbf{x})$[12] for CRFs is given by (Lafferty et al., 2001)

$$P(\mathbf{y}|\mathbf{x}) = \frac{\prod_i \phi(C_i, \mathbf{x})}{\sum_{\mathbf{y}} \prod_i \phi(C_i, \mathbf{x})}$$

where $\phi(C_i, \mathbf{x})$ is a potential function defined over each clique C_i. Potential functions are typically parameterized in an exponential form as follows.

$$\phi(C_i, \mathbf{x}) = \exp \sum_k \lambda_k f_k(C_i, \mathbf{x})$$

where λ_k are the parameters and $f_k(\cdot)$ are feature indicator functions. Because the Hammersley-Clifford theorem (1971) for undirected graphical models holds for any non-negative potential functions, we propose alternative potential functions as follows.

$$\psi(C_i, \mathbf{x}) = \begin{cases} \beta\phi(C_i, \mathbf{x}) & \text{if } \mu(C_i, \mathbf{x}) = true \\ \phi(C_i, \mathbf{x}) & \text{otherwise} \end{cases}$$

where β is a non-negative bias factor, and $\mu(C_i, \mathbf{x})$ is a predicate (or an indicator function) to check certain properties on $(C_i, \mathbf{x})$.[13] Examples of possible $\mu(\cdot)$ would be whether the true assignment for C_i in the training data contains certain class values, or whether the current observation indexed by C_i has particular characteristics. More specific details will be given in §4.2.

Training and testing with biased potential functions is mostly identical to the traditional log-linear formulations by $\phi(\cdot)$ as defined above, except for small and straightforward modifications to the computation of the likelihood and the derivative of the likelihood.

[12]For the local model described in Section 2, **y** should be replaced with **h**. We use **y** in this section however, as it is a more conventional notation in general.

[13]In our problem formulation, cliques are individual nodes, and potential functions are defined over the observations indexed by the current i only: i.e. $\phi(C_i, \mathbf{x}) = \phi(y_i, x_i)$, $\mu(C_i, \mathbf{x}) = \mu(y_i, x_i)$ and $\psi(C_i, \mathbf{x}) = \psi(y_i, x_i)$.

The key idea for biased potential functions is nothing new, as it is conceptually similar to instance weighting for problems with non-structured output (e.g. Aha and Goldstone (1992), Cardie et al. (1997)). However, biased potential functions differ technically in that they emphasize desired subcomponents without altering the i.i.d. assumption, and still weight each instance alike. Despite the conceptual simplicity, we are not aware of any previous work that explored biased potential functions for problems with structured output.

4.2 Applications to Coreference Resolution

[Bias on Coreferent Pairs] For coreference resolution, pairs that are coreferent are in a minority class[14], and biased potential functions can mitigate this skewed data problem, by amplifying the clique potentials that correspond to coreferent pairs. We define $\mu(y_i, x_i)$ to be true if and only if the true assignment for y_i in the training data is *'coreferent'*. Notice that $\mu(\cdot)$ does not depend on what particular value y_i might take, but only depends on the true value of y_i in the training data. For testing, $\mu(y_i, x_i)$ will be always false.[15]

[Bias on Closer Coreferent Pairs] For coreference resolution, we hypothesize that coreferent pairs for closer mentions have more significance, because they tend to have clearer linguistic clues to determine coreference. We further hypothesize that by emphasizing only close coreferent pairs, we can have our model favor the MUC score. For this, we define $\mu(y_i, x_i)$ to be true if and only if x_i is for a pair of mentions that are the closest coreferent pair.

5 Experiments–II

Data sets and configurations for experiments are identical to those used in §3.

Hypothesis: We hypothesize that using biased potential functions, maximizing the likelihood for training can correlate better with F1-score or MUC-score than the pairwise accuracy. In particular,

[14]Only 1.72% of the pairs are coreferent in the MUC6 data, and about 12% are coreferent in the MPQA data.

[15]Notice that $\mu(y_i, x_i)$ changes the surface of the likelihood for training, but does not affect the inference of finding the argmax in our local model. That is, $\text{argmax}_{y_i} \phi(y_i, x_i) = \text{argmax}_{y_i} \psi(y_i, x_i)$ (with y_i replaced with h_i).

MUC6	pairwise				MUC		
	e %	R %	P %	F %	R %	P %	F %
BASE	1.18	38.0	**85.6**	52.6	59.0	**75.8**	66.4
BASIC-P1$^{1.5}$	1.20	38.9	82.1	52.8	64.2	71.8	**67.8**
BASIC-P1$^{3.0}$	1.32	46.9	71.3	56.6	68.9	64.3	66.5
BASIC-P$a^{1.5}$	**1.15**	44.2	79.9	56.9	62.1	68.7	65.2
BASIC-P$a^{3.0}$	1.44	**52.5**	62.9	**57.2**	**70.9**	60.5	65.3

MPQA	pairwise				MUC		
	e %	R %	P %	F %	R %	P %	F %
BASE	**7.05**	52.1	**83.4**	64.1	75.6	**81.5**	**78.4**
BASIC-P1$^{1.5}$	7.18	54.6	79.6	64.8	77.7	76.5	77.1
BASIC-P1$^{3.0}$	7.22	59.9	75.4	66.8	83.3	71.7	77.1
BASIC-P$a^{1.5}$	7.65	59.7	72.2	65.4	79.8	73.2	76.4
BASIC-P$a^{3.0}$	8.22	**69.2**	65.1	**67.1**	**85.8**	67.8	75.7

Table 2: Performance of Biased Potential Functions: pairwise scores are taken before single-link-clustering is applied.

MUC6	pairwise				MUC		
	e %	R %	P %	F %	R %	P %	F %
BASE	1.50	59.2	56.2	57.7	59.0	75.8	66.4
SLT	1.28	49.8	67.3	57.2	56.3	**77.8**	65.3
SLT-P1$^{1.5}$	**1.19**	52.8	**70.6**	60.4	59.3	74.6	66.1
SLT-P1$^{3.0}$	1.42	63.5	57.9	**60.6**	67.5	70.7	69.1
SLT-P$a^{1.5}$	1.43	58.6	58.5	58.5*	64.0	73.6	68.5
SLT-P$a^{3.0}$	1.71	**65.2**	50.3	56.8	**70.5**	69.3	**69.9***

MPQA	pairwise				MUC		
	e %	R %	P %	F %	R %	P %	F %
BASE	9.83	75.8	57.0	65.1	75.6	81.5	78.4
SLT	**6.39**	62.1	**80.6**	70.2	69.1	**88.2**	77.5
SLT-P1$^{1.5}$	6.54	64.9	77.4	**70.6***	72.2	84.5	77.9*
SLT-P1$^{3.0}$	9.09	77.2	59.6	67.3	78.4	79.5	78.9
SLT-P$a^{1.5}$	6.74	65.2	75.7	70.1	72.4	87.2	**79.1**
SLT-P$a^{3.0}$	14.71	**78.2**	43.9	56.2	**80.5**	73.8	77.0

Table 3: Performance of Biased Potential Functions with Structured Local Training: All numbers are taken after single-link clustering.

we hypothesize that biasing on every coreferent pair will correlate more with F1-score, and biasing on close coreferent pairs will correlate more with MUC-score. In general, we expect that biasing on coreferent pairs will boost recall, potentially decreasing precision.

Results [BPF]: Experimental results for biased potential functions, without structured local training, are shown in Table 2. BASIC-P1$^{\beta}$ denotes local training with biased potential on the closest coreferent pairs with bias factor β, and BASIC-Pa^{β} denotes local training with biased potential on the all coreferent pairs with bias factor β, where $\beta = 1.5$ or 3.0. For brevity, we only show pairwise numbers before applying single-link-clustering.[16] As hypothesized, biased potential functions in general boost recall at the cost of precision. Also, for a fixed value of β, BASIC-P1$^{\beta}$ gives better MUC-F1 than BASIC-Pa^{β}, and BASIC-Pa^{β} gives better pairwise-F1 than BASIC-P1$^{\beta}$ for both data sets.

Results [SLT+BPF]: Experimental results that combine SLT and BPF are shown in Table 3. Similarly as before, SLT-Px^{β} denotes SLT with biased potential scheme Px, with bias factor β. For brevity, we only show numbers after applying single-link-clustering. Unlike the results shown in Table 2, for a fixed value of β, SLT-P1$^{\beta}$ correlates better with pairwise-F1, and SLT-Pa^{β} correlates better with MUC-F1. This indicates that when biased potential functions are used in conjunction with SLT, the effect of biased potential functions can be different from the case without SLT. Comparing F1-scores in Table 2 and Table 3, we see that the combination of biased potential functions with SLT improves performance in general. In particular, SLT-P1$^{3.0}$ and SLT-P$a^{1.5}$ consistently improve performance over BASE on both data sets, for both pairwise-F1 and MUC-F1. We present performance scores for all variations of configurations for reference, but we also mark the particular configuration SLT-Px^{β} (by '*' on F1-scores) that is chosen when selecting the configuration based on the performance on the training data for each performance measure. To conclude, structured local training with biased potential functions bring a substantial improvement for MUC-F1 score, from 66.4% to 69.9% for MUC6 data set. For pairwise-F1, the performance increase from 57.7% to 58.5% for MUC6, and from 65.1% to 70.6% for MPQA.[17]

[16]This is because we showed in §3 that basic local training does not correlate well with pairwise scores after clustering, and in order to see the direct effect of biased potential functions, we examine pairwise numbers before clustering.

[17]Performance on the MPQA data for MUC-F1 is slightly decreased from 78.4% to 77.9%. Note the MUC scores for the

6 Related Work

Structured local training is motivated by recent research that has shown that reducing the discrepancy between the training model and testing model can improve the performance without incurring the heavy computational overhead of full-blown global inference-based training.[18] (e.g. Cohen and Carvalho (2005), Sutton and McCallum (2005a), Sutton and McCallum (2005b)). Our work differs in that (1) we use hidden variables to capture the interactions between local inference and global inference, (2) we present an application to coreference resolution, while previous work has shown applications for variants of sequence tagging. McCallum and Wellner (2004) showed a global training approach with CRFs for coreference resolution, but they used the voted perceptron algorithm for training, which no longer maximizes the likelihood. In addition, they assume that all and only those noun phrases involved in coreference resolution are given.

The performance of our system on MUC6 data set is comparable to previously reported systems. Using the same feature set, Ng and Cardie (2002) reports 64.5% of MUC-score, while our system achieved 69.9%. Ng and Cardie (2002) reports 70.4% of MUC-score using hand-selected features. With an additional feature selection or feature induction step, the performance of our system might further improve. McCallum and Wellner (2004) reports 73.42% of MUC-score on MUC6 data set, but their experiments assumed perfect identification of all and only those noun phrases involved in a coreference relation, thus substantially simplifying the task.

7 Conclusion

We present a novel training procedure, *structured local training*, that maximizes likelihood while exploiting the benefits of global inference during training. This is achieved by incorporating hidden variables to capture the interactions between local inference and global inference. In addition, we introduce *biased potential functions* that allow CRFs to empirically favor performance measures such as F1-score or MUC-score. We focused on the application of coreference resolution in this paper, but the key ideas of our approaches can be extended to other applications, and other machine learning techniques motivated by Markov networks.

MPQA baseline are already quite high to begin with.

[18]The computational cost for SLT in our experiments were about twice of the cost for the local training of the baseline. This is the case because M-step converges very fast from the second EM iteration, by initializing CRFs using parameters from the previous M-step. Biased potential functions hardly adds extra computational cost. In practice, BPFs reduce training time substantially: we observed that the higher the bias is, the quicker CRFs converge.

Acknowledgments We thank the reviewers as well as Eric Breck and Ves Stoyanov for their many helpful comments. This work was supported by the Advanced Research and Development Activity (ARDA), by NSF Grants BCS-0624277, IIS-0535099, and IIS-0208028, and by gifts from Google and the Xerox Foundation.

References

D.W. Aha and R.L. Goldstone. 1992. Concept learning and flexible weighting. In *Proc. of the Fourteenth Annual Conference of the Cognitive Science Society*.

A. Berger, S.D. Pietra, V.D. Pietra 1996. A Maximum Entropy Approach to Natural Language Processing. In *Computational Linguistics*,22.

C. Cardie and N. Howe. 1997. Improving Minority Class Prediction Using Case-Specific Feature Weights. In *ICML*.

W.W. Cohen and V. Carvalho. 2005. Stacked Sequential Learning. In *IJCAI*.

M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP*.

A.P. Dempster, N. M. Laird and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. In *Journal of the Loyal Statistical Society*, B.39.

J. Finkel, T. Grenager and C. D. Manning. 2005. Incorporating Non-local Information Into Information Extraction Systems By Gibbs Sampling. In *ACL*.

T. Finley and T. Joachims. 2005. Supervised Clustering with Support Vector Machines. In *ICML*.

J. Hammersley and P. Clifford. 1971. Markov fields on finite graphs and lattices. Unpublished manuscript.

J. Lafferty, A. McCallum and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*.

A. McCallum and B. Wellner. 2004. Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *NIPS*.

MUC-6 1995. In Proc. of the Sixth Message Understanding Conference (MUC-6) Morgan Kaufmann.

R. M. Neal and G. E. Hinton. 1998. A view of the EM algorithm that justies incremental, sparse, and other variants. In *Learning in Graphical Models*, Kluwer.

V. Ng and C. Cardie. 2002. Improving Machine Learning Approaches to Coreference Resolution. In *ACL*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak 2005. Learning and Inference over Constrained Output. In *IJCAI*.

D. Roth and W. Yih. 2005. Integer Linear Programming Inference for Conditional Random Fields. In *ICML*.

V. Stoyanov and C. Cardie. 2006. Partially Supervised Coreference Resolution for Opinion Summarization through Structured Rule Learning. In *EMNLP*.

C. Sutton and A. McCallum. 2005. Fast, Piecewise Training for Discriminative Finite-state and Parsing Models. In *Technical Report IR-403, University of Massachusetts*.

C. Sutton and A. McCallum. 2005. Piecewise Training for Undirected Models. In *UAI*.

B. Wellner, A. McCallum, F. Peng and M. Hay. 2004. An Integrated, Conditional Model of Information Extraction and Coreference with Application to Citation Matching. In *UAI*.

J. Wiebe and T. Wilson and C. Cardie 2005. Annotating Expressions of Opinions and Emotions in Language. In *Language Resources and Evaluation, volume 39, issue 2-3*.

Coreference or Not: A Twin Model for Coreference Resolution

Xiaoqiang Luo
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, U.S.A.
{xiaoluo}@us.ibm.com

Abstract

A twin-model is proposed for coreference resolution: a *link* component, modeling the coreferential relationship between an anaphor and a candidate antecedent, and a *creation* component modeling the possibility that a phrase is not coreferential with any candidate antecedent. The creation model depends on all candidate antecedents and is often expensive to compute; Therefore constraints are imposed on feature forms so that features in the creation model can be efficiently computed from feature values in the link model. The proposed twin-model is tested on the data from the 2005 Automatic Content Extraction (ACE) task and the proposed model performs better than a thresholding baseline without tuning free parameter.

1 Introduction

Coreference resolution aims to find multiple mentions of an entity (e.g., PERSON, ORGANIZATION) in a document. In a typical machine learning-based coreference resolution system (Soon et al., 2001; Ng and Cardie, 2002b; Yang et al., 2003; Luo et al., 2004), a statistical model is learned from training data and is used to measure how likely an anaphor [1] is coreferential to a candidate antecedent. A related, but often overlooked, problem is that the anaphor may be non-coreferential to any candidate, which arises from scenarios such as an identified anaphor is truly generic and there does not exist an antecedent in the discourse context, or an anaphor is the first mention (relative to processing order) in a coreference chain.

In (Soon et al., 2001; Ng and Cardie, 2002b), the problem is treated by thresholding the scores returned by the coreference model. That is, if the maximum coreference score is below a threshold, then the anaphor is deemed non-referential to any candidate antecedent. The threshold approach does not model non-coreferential events directly, and is by no means the optimal approach to the problem. It also introduces a free parameter which has to be set by trial-and-error. As an improvement, Ng and Cardie (2002a) and Ng (2004) train a separate model to classify an anaphor as either *anaphoric* or *non-anaphoric*. The output of this classifier can be used either as a pre-filter (Ng and Cardie, 2002a) so that *non-anaphoric* anaphors will not be precessed in the coreference system, or as a set of features in the coreference model (Ng, 2004). By rejecting any anaphor classified as *non-anaphoric* in coreference resolution, the filtering approach is meant to handle non-anaphoric phrases (i.e., no antecedent exists in the discourse under consideration), not the first mention in a coreference chain.

In this paper, coreference is viewed as a process of sequential operations on anaphor mentions: an anaphor can either be *linked* with its antecedent if the antecedent is available or present. If the anaphor, on the other hand, is discourse new (relative to the process order), then a new entity is *created*. Corresponding to the two types of operations, a twin-model is proposed to resolve coreferential relationships in a document. The first component is a statistical model measuring how likely an anaphor is coreferential to a candidate antecedent; The second one explicitly models the non-

[1] In this paper, "anaphor" includes all kinds of phrases to be resolved, which can be named, nominal or pronominal phrases.

Proceedings of NAACL HLT 2007, pages 73–80,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

coreferential events. Both models are trained automatically and are used simultaneously in the coreference system. The twin-model coreference system is tested on the 2005 ACE (Automatic Content Extraction, see (NIST, 2005)) data and the best performance under both ACE-Value and entity F-measure can be obtained without tuning a free parameter.

The rest of the paper is organized as follows. The twin-model is presented in Section 2. A maximum-entropy implementation and features are then presented in Section 3. The experimental results on the 2005 ACE data is presented in Section 4. The proposed twin-model is compared with related work in Section 5 before the paper is concluded.

2 Coreference Model

A phrasal reference to an entity is called a mention. A set of mentions referring to the same physical object is said to belong to the same entity. For example, in the following sentence:

(I) `John` said `Mary` was `his` `sister`.

there are four mentions: `John`, `Mary`, `his`, and `sister`. `John` and `his` belong to the same entity since they refer to the same person; So do `Mary` and `sister`. Furthermore, `John` and `Mary` are *named* mentions, `sister` is a *nominal* mention and `his` is a *pronominal* mention.

In our coreference system, mentions are processed sequentially, though not necessarily in chronological order. For a document with n mentions $\{m_i : 1 \leq i \leq n\}$, at any time $t (t > 1)$, mention m_1 through m_{t-1} have been processed and each mention is placed in one of $N_t (N_t \leq (t-1))$ entities: $E_t = \{e_j : 1 \leq j \leq N_t\}$. Index i in m_i indicates the order in which it is processed, not necessarily the order in which it appears in a document. The basic step is to extend E_t to E_{t+1} with m_t.

Let us use the example in Figure 1 to illustrate how this is done. Note that Figure 1 contains one possible processing order for the four mentions in Example (I): first name mentions are processed, followed by nominal mentions, followed by pronominal mentions. At time $t = 1$, there is no existing entity and the mention m_1=`John` is placed in an initial entity (entity is signified by a solid rectangle). At time $t = 2$, m_2=`Mary` is processed and a new entity containing `Mary` is created. At time $t = 3$, the nominal mention m_3=`sister` is processed. At this point, the set of existing entities

$$E_3 = \Big\{\{\texttt{John}\}, \{\texttt{Mary}\}\Big\}.$$

m_3 is linked with the existing entity $\{\texttt{Mary}\}$. At the last step $t = 4$, the *pronominal* mention `his` is linked with the entity $\{\texttt{John}\}$.

The above example illustrates how a sequence of coreference steps lead to a particular coreference result. Conversely, if the processing order is known and fixed, every possible coreference result can be decomposed and mapped to a unique sequence of such coreference steps. Therefore, if we can score the set of coreference sequences, we can score the set of coreference results as well.

In general, when determining if a mention m_t is coreferential with any entity in E_t, there are two types of actions: one is that m_t is coreferential with one of the entities; The other is that m_t is not coreferential with any. It is important to distinguish the two cases for the following reason: if m_t is coreferential with an entity e_j, in most cases it is sufficient to determine the relationship by examining m_t and e_j, and their local context; But if m_t is not coreferential with any existing entities, we need to consider m_t with all members in E_t. This observation leads us to propose the following twin-model for coreference resolution.

The first model, $P(L|e_j, m_t)$, is conditioned on an entity e_j and the current mention m_t and measure how likely they are coreferential. L is a binary variable, taking value 1 or 0, which represents positive and negative coreferential relationship, respectively. The second model, on the other hand, $P(C|E_t, m_t)$, is conditioned on the past entities E_t and the current mention m_t. The random variable C is also binary: when C is 1, it means that a new entity $\{m_t\}$ will be created. In other words, the second model measures the probability that m_t is *not* coreferential to any existing entity. To avoid confusion in the subsequent presentation, the first model will be written as $P_l(\cdot|e_j, m_t)$ and called *link* model; The second model is written as $P_c(\cdot|E_t, m_t)$ and called *creation* model.

For the time being, let's assume that we have the link and creation model at our disposal, and we will show how they can be used to score coreference decisions.

Given a set of existing entities $E_t = \{e_j\}_1^{N_t}$, formed by mentions $\{m_i\}_{i=1}^{t-1}$, and the current mention m_t, there are $N_t + 1$ possible actions: we can either link m_t with an existing entity e_j $(j = 1, 2, \cdots, N_t)$, or create a new entity containing m_t. The link action between e_j and m_t can be scored by $P_l(1|e_j, m_t)$ while the creation action can be measured by $P_c(1|E_t, m_t)$. Each possible coreference outcome consists of n such actions $\{a_t : t = 1, 2, \cdots, n\}$, each of which can be scored by either the link model $P_l(\cdot|e_j, m_t)$ or the cre-

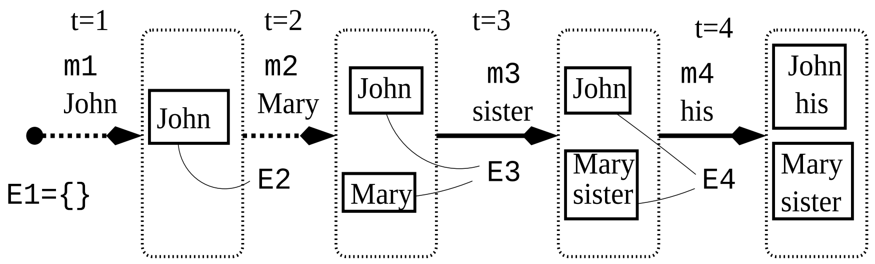


Figure 1: Coreference process for the four mentions in Example (I). Mentions in a document are processed sequentially: first name mentions, then nominal mentions, and then pronominal mentions. A dashed arrow signifies that a new entity is created, while a solid arrow means that the current mention is linked with an existing entity.

ation model $P_c(\cdot|E_t, m_t)$. Denote the score for action a_t by $S(a_t|a_1^{t-1})$, where dependency of a_t on a_1 through a_{t-1} is emphasized. The coreference result corresponding to the action sequence is written as $E_n(\{a_i\}_{i=1}^n)$. When it is clear from context, we will drop $\{a_i\}_{i=1}^n$ and write E_n only.

With this notation, the score for a coreference outcome $E_n(\{a_i\}_{i=1}^n)$ is the product of individual scores assigned to the corresponding action sequence $\{a_i\}_{i=1}^n$, and the best coreference result is the one with the highest score:

$$\begin{aligned}\hat{E}_n &= arg\max_{E_n} S(E_n)\\ &= arg\max_{\{a_t\}_1^n} \prod_{t=1}^{n} S(a_t|a_1^{t-1}). \qquad (1)\end{aligned}$$

Given n mentions, the number of all possible entity outcomes is the Bell Number (Bell, 1934): $B(n) = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$. Exhaustive search is out of the question. Thus, we organize hypotheses into a Bell Tree (Luo et al., 2004) and use a beam search with the following pruning strategy: first, a maximum beam size (typically 20) S is set, and we keep only the top S hypotheses; Second, a relative threshold r (we use 10^{-5}) is set to prune any hypothesis whose score divided by the maximum score falls below the threshold.

To give an concrete example, we use the example in Figure 1 again. The first step at $t = 1$ creates a new entity and is therefore scored by $P_c(1|\{\}, \texttt{John})$; the second step also creates an entity and is scored by $P_c(1|\{\texttt{John}\}, \texttt{Mary})$; the step $t = 3$, however, links `sister` with $\{\texttt{Mary}\}$ and is scored by $P_l(1|\{\texttt{Mary}\}, \texttt{sister})$; Similarly, the last step is scored by $P_l(1|\{\texttt{John}\}, \texttt{his})$. The score for this coreference outcome is the product of the four numbers:

$$\begin{aligned}&S(\Big\{\{\texttt{John,his}\}, \{\texttt{Mary,sister}\}\Big\})\\ =&P_c(1|\{\}, \texttt{John})P_c(1|\{\texttt{John}\}, \texttt{Mary})\cdot\\ &P_l(1|\{\texttt{Mary}\}, \texttt{sister})\cdot\\ &P_l(1|\{\texttt{John}\}, \texttt{his}). \qquad (2)\end{aligned}$$

Other coreference results for these four mentions can be scored similarly. For example, if `his` at the last step is linked with $\{\texttt{Mary,sister}\}$, the score would be:

$$\begin{aligned}&S(\Big\{\{\texttt{John}\}, \{\texttt{Mary,sister,his}\}\Big\})\\ =&P_c(1|\{\}, \texttt{John})P_c(1|\{\texttt{John}\}, \texttt{Mary})\cdot\\ &P_l(1|\{\texttt{Mary}\}, \texttt{sister})\cdot\\ &P_l(1|\{\texttt{Mary,sister}\}, \texttt{his}). \qquad (3)\end{aligned}$$

At testing time, (2) and (3), among other possible outcomes, will be searched and compared, and the one with the highest score will be output as the coreference result.

Examples in (2) and (3) indicate that the link model $P_l(\cdot|e_j, m_t)$ and creation model $P_c(\cdot|E_t, m_t)$ form an integrated coreference system and are applied simultaneously at testing time. As will be shown in the next section, features in the creation model $P_c(\cdot|E_t, m_t)$ can be computed from their counterpart in the link model $P_l(\cdot|e_j, m_t)$ under some mild constraints. So the two models' training procedures are tightly coupled. This is different from (Ng and Cardie, 2002a; Ng, 2004) where their anaphoricty models are trained independently of the coreference model, and it is either used as a pre-filter, or its output is used as features in the coreference model. The creation model $P_c(\cdot|E_t, m_t)$ proposed here bears similarity to the *starting* model in (Luo et al., 2004). But there is a crucial difference: the *starting* model in (Luo et al., 2004) is an ad-hoc use of the link scores and is not learned automatically, while $P_c(\cdot|E_t, m_t)$ is fully trained. Training $P_c(\cdot|E_t, m_t)$ is covered in the next section.

3 Implementation

3.1 Feature Structure

To implement the twin model, we adopt the log linear or maximum entropy (MaxEnt) model (Berger et al., 1996) for its flexibility of combining diverse sources of information. The two models are of the form:

$$P_l(L|e_j, m_t) = \frac{exp\Big(\sum_k \lambda_k g_k(e_j, m_t, L)\Big)}{Y(e_j, m_t)} \quad (4)$$

$$P_c(C|E_t, m_t) = \frac{exp\Big(\sum_i \nu_i h_i(E_t, m_t, C)\Big)}{Z(E_t, m_t)}, \quad (5)$$

where L and C are binary variables indicating either m_t is coreferential with e_j, or m_t is used to create a new entity. $Y(e_j, m_t)$ and $Z(e_j, m_t)$ are normalization factors to ensure that $P_l(\cdot|e_j, m_t)$ and $P_c(\cdot|E_t, m_t)$ are probabilities; λ_k and ν_i are the weights for feature $g_k(e_j, m_t, L)$ and $h_i(E_t, m_t, C)$, respectively. Once the set of features functions are selected, algorithm such as improved iterative scaling (Berger et al., 1996) or sequential conditional generalized iterative scaling (Goodman, 2002) can be used to find the optimal parameter values of $\{\lambda_k\}$ and $\{\nu_i\}$.

Computing features $\{g_k(e_j, m_t, \cdot)\}$ for the link model $P_l(L|e_j, m_t)$ [2] is relatively straightforward: given an entity e_j and the current mention m_t, we just need to characterize things such as lexical similarity, syntactic relationship, and/or semantic compatibility of the two. It is, however, very challenging to compute the features $\{h_i(E_t, m_t, \cdot)\}$ for the creation model $P_c(\cdot|E_t, m_t)$ since its conditioning includes a set of entities E_t, whose size grows as more and more mentions are processed. The problem exists because the decision of creating a new entity with m_t has to be made after examining all preceding entities. There is no reasonable modeling assumption one can make to drop some entities in the conditioning.

To overcome the difficulty, we impose the following constraints on the features of the link and creation model:

$$g_k(e_j, m_t, L) = g_k^{(1)}(e_j, m_t) g_k^{(2)}(L) \quad (6)$$

$$h_i(E_t, m_t, C) = h_i^{(1)}\big(\{g_k^{(1)}(e, m_t) : e \in E_t\}\big) \cdot h_i^{(2)}(C), \text{ for some } k. \quad (7)$$

(6) states that a feature in the link model is separable and can be written as a product of two functions: the first one, $g_k^{(1)}(\cdot, \cdot)$, is a binary function depending on the conditioning part only; the second one, $g_k^{(2)}(\cdot)$, is an indicator function depending on the prediction part L only. Like $g_k^{(2)}(\cdot)$, $h_i^{(2)}(\cdot)$ is also a binary indicator function.

(7) implies that features in the creation model are also separable; Moreover, the conditioning part $h_i^{(1)}\big(\{g_k^{(1)}(e, m_t) : e \in E_t\}\big)$, also a binary function, only depends on the function values of the set of link features $\{g_k^{(1)}(e, m_t) : e \in E_t\}$ (for some k). In other words, once $\{g_k^{(1)}(e, m_t) : e \in E_t\}$ and C are known, we can compute $h_i(E_t, m_t, C)$ without actually comparing m_t with any entity in E_t. Using binary features is a fairly mild constraint as non-binary features can be replaced by a set of binary features through quantization.

How fast $h_i^{(1)}\big(\{g_k^{(1)}(e, m_t) : e \in E_t\}\big)$ can be computed depends on how $h_i^{(1)}$ is defined. In most cases – as will be shown in Section 3.2, it boils down testing if any member in $\{g_k^{(1)}(e, m_t) : e \in E_t\}$ is non-zero; or counting how many non-zero members there are in $\{g_k^{(1)}(e, m_t) : e \in E_t\}$. Both are simple operations that can be carried out quickly. Thus, the assumption (7) makes it possible to compute efficiently $h_i(E_t, m_t, C)$.

3.2 Features in the Creation Model

We describe features used in our coreference system. We will concentrate on features used in the creation model since those in the link model can be found in the literature (Soon et al., 2001; Ng and Cardie, 2002b; Yang et al., 2003; Luo et al., 2004). In particular, we show how features in the creation model can be computed from a set of feature values from the link model for a few example categories. Since $g_k^{(2)}(\cdot)$ and $h_i^{(2)}(\cdot)$ are simple indicator functions, we will focus on $g_k^{(1)}(\cdot, \cdot)$ and $h_i^{(1)}(\cdot)$.

3.2.1 Lexical Features

This set of features computes if two surface strings (spellings of two mentions) match each other, and are

[2] The link model is actually implemented as: $P_l(L|e_j, m_t) \approx \max_{m' \in e_j} \hat{P}_l(L|e_j, m', m_t)$. Some features are computed on a pair of mentions (m', m_t) while some are computed at entity level. See (Luo and Zitouni, 2005) and (Daumé III and Marcu, 2005).

applied to name and nominal mentions only. For the link model, a lexical feature $g_k^{(1)}(e_j, m_t)$ is 1 if e_j contains a mention matches m_t, where a match can be exact, partial, or one is an acronym of the other.

Since $g_k(e_j, m_t)$ is binary, one corresponding feature used in the creation model is the disjunction of the values in the link model, or

$$h_i^{(1)}(E_t, m_t) = \vee_{e \in E_t}\{g_k^{(1)}(e, m_t)\}, \qquad (8)$$

where $\vee$ is a binary "or" operator. The intuition is that if there is any mention in E_t matching m_t, then the probability to create a new entity with m_t should be low; Conversely, if none of the mentions in E_t matches m_t, then m_t is likely to be the first mention of a new entity.

Take $t = 2$ in Figure 1 as an example. There is only one partially-established entity $\{\texttt{John}\}$, so $E_2 = \{\texttt{John}\}$, and $m_2 = \texttt{Mary}$. The exact string match feature $g_{em}^{(1)}(\cdot,\cdot)$ would be

$$g_{em}^{(1)}(\{\texttt{John}\}, \texttt{Mary}) = 0,$$

and the corresponding string match feature in the creation model is

$$\begin{aligned} h_{em}^{(1)}(\{\texttt{John}\}, \texttt{Mary}) &= \vee_{e \in E_t}\{g_{em}^{(1)}(e, \texttt{Mary})\} \\ &= 0. \end{aligned}$$

Disjunction is not the only operation we can use. Another possibility is counting how many times m_t matches mentions in E_t, so (8) becomes:

$$h_i^{(1)}(E_t, m_t) = Q\big[\sum_{e \in E_t}\{g_k^{(1)}(e, m_t)\}\big], . \qquad (9)$$

where $Q[\cdot]$ quantizes raw counts into bins.

3.2.2 Attribute Features

In the link model, features in this category compare the properties of the current mention m_t with that of an entity e_j. Properties of a mention or an entity, whenever applicable, include gender, number, entity type, reflexivity of pronouns etc. Similar to what done in the lexical feature, we again synthesize a feature in the creation model by taking the disjunction of the corresponding set of feature values in the link model, or

$$h_i^{(1)}(E_t, m_t) = \vee_{e \in E_t}\{g_k^{(1)}(e, m_t)\},$$

where $g_k^{(1)}(e, m_t)$ takes value 1 if entity e and mention m_t share the same property; Otherwise its value is 0. The intuition is that if there is an entity having the same property as the current mention, then the probability for the current mention to be linked with the entity should be higher than otherwise; Conversely, if none of the entities in E_t shares a property with the current mention, the probability for the current mention to create a new entity ought to be higher.

Consider the gender attribute at $t = 4$ in Figure 1. Let $g_{gender}^{(1)}(\cdot,\cdot)$ be the gender feature in the link model, assume that we know the gender of `John`, `Mary` and `his`. Then $g_{gender}^{(1)}(\{\{\texttt{John}\}, \texttt{his})$ is 1, while $g_{gender}^{(1)}(\{\texttt{Mary, sister}\}, \texttt{his})$ is 0. Therefore, the gender feature for the creation model would be

$$\begin{aligned} &h_{gender}^{(1)}(\{\{\texttt{John}\},\{\texttt{Mary, sister}\}\}, \texttt{his}) \\ &= 0 \vee 1 = 1, \end{aligned}$$

which means that there is at least one mention which has the same the gender of the current mention m_t.

3.2.3 Distance Feature

Distance feature needs special treatment: while it makes sense to talk about the distance between a pair of mentions, it is not immediately clear how to compute the distance between a set of entities E_t and a mention m_t. To this end, we compute the minimum distance between the entities and the current mention with respect to a "fired" link feature, as follows.

For a particular feature $g_k^{(1)}(\cdot,\cdot)$ in the link model, define the minimum distance to be

$$\begin{aligned} \hat{d}(E_t, m_t; g_k) = \min\{&d(m, m_t) : m \in E_t, \\ &\text{and } g_k^{(1)}(m, m_t) = 1\}, \qquad (10) \end{aligned}$$

where $d(m, m_t)$ is the distance between mention $m$and m_t. The distance itself can be the number of tokens, or the number of intervening mentions, or the number of sentences. The minimum distance $\hat{d}(E_t, m_t; g_k)$ is quantized and represented as binary feature in the creation model. The idea here is to encode what is the nearest place where a feature fires.

Again as an example, consider the gender attribute at $t = 4$ in Figure 1. Assuming that $d(m, m_t)$ is the number of tokens. Since only `John` matches the gender of `his`,

$$\hat{d}(E_4, m_4; g_{gender}) = 3.$$

The number is then quantized and used as a binary feature to encode the information that "there is a mention whose gender matches the current mention within in a token distance range including 3."

In general, binary features in the link model which measure the similarity between an entity and a mention can be turned into features in the creation model in the same manner as described in Section 3.2.1 and 3.2.2. For example, syntactic features (Ng and Cardie, 2002b; Luo and Zitouni, 2005) can be computed this way and are used in our system.

4 Experiments

4.1 Data and Evaluation Metric

We report the experimental results on ACE 2005 data (NIST, 2005). The dataset consists of 599 documents from a rich and diversified sources, which include newswire articles, web logs, and Usenet posts, transcription of broadcast news, broadcast conversations and telephone conversations. We reserve the last 16% documents of each source as the test set and use the rest of the documents as the training set. Statistics such as the number of documents, words, mentions and entities of this data split is tabulated in Table 1.

DataSet	#Docs	#Words	#Mentions	#Entities
Training	499	253771	46646	16102
Test	100	45659	8178	2709
Total	599	299430	54824	18811

Table 1: Statistics of ACE 2005 data: number of documents, words, mentions and entities in the training and test set.

The link and creation model are trained at the same time. Besides the basic feature categories described in Section 3.2, we also compute composite features by taking conjunctions of the basic features. Features are selected by their counts with a threshold of 8.

ACE-Value is the official score reported in the ACE task and will be used to report our coreference system's performance. Its detailed definition can be found in the official evaluation document [3]. Since ACE-Value is a weighted metric measuring a coreference system's relative value, and it is not sensitive to certain type of errors (e.g., false-alarm entities if these entities contain correct mentions), we also report results using unweighted entity F-measure.

4.2 Results

To compare the proposed twin model with simple thresholding (Soon et al., 2001; Ng and Cardie, 2002b),

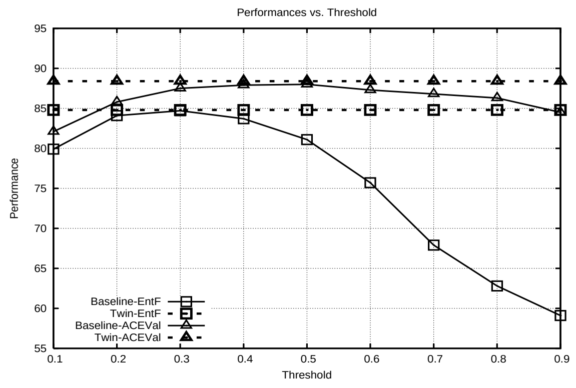


Figure 2: Performance comparison between a thresholding baseline and the twin-model: lines with square points are the entity F-measure (x100) results; lines with triangle points are ACE-Value (in %). Solid lines are baseline while dashed lines are twin-model.

we first train our twin model. To simulate the thresholding approach, a baseline coreference system is created by replacing the creation model with a constant, i.e.,

$$P_c(1|E_t, m_t) = \theta, \tag{11}$$

where θ is a number between 0 and 1. At testing time, a new entity is created with score θ when

$$P_l(1|e_j, m_t) < \theta, \ \ \forall e_j \in E_t.$$

The decision rule simply implies that if the scores between the current mention m_t and all candidate entities $e_j \in E_t$ are below the threshold θ, a new entity will be created.

Performance comparison between the baseline and the twin-model is plotted in Figure 2. X-axis is the threshold varying from 0.1 to 0.9 with a step size 0.1. Two metrics are used to compare the results: two lines with square data points are the entity F-measure results, and two lines with triangle points are ACE-Value. Note that performances for the twin-model are constant since it does not use thresholding.

As shown in the graph, the twin-model (two dashed lines) always outperforms the baseline (two solid lines). A "bad" threshold impacts the entity F-measure much more than ACE-Value, especially in the region with high threshold value. Note that a large θ will lead to more false-alarm entities. The graph suggests that ACE-Value is much less sensitive than the un-weighted F-measure in measuring false-alarm errors. For example, at $\theta = 0.9$, the baseline F-measure is 0.591 while

[3]The offi cial evaluation document can be found at: www.nist.gov/speech/tests/ace/ace05/doc/ace05-evalplan.v3.pdf.

the twin model F-measure is 0.848, a 43.5% difference; On the other hand, the corresponding ACE-Values are 84.5% (baseline) vs. 88.4% (twin model), a mere 4.6% relative difference. There are at least two reasons: first, ACE-Value discounts importance of nominal and pronoun entities, so more nominal and pronoun entity errors are not reflected in the metric; Second, ACE-Value does not penalize false-alarm entities if they contain correct mentions. The problem associated with ACE-Value is the reason we include the entity F-measure results.

Another interesting observation is that an optimal threshold for the entity F-measure is not necessarily optimal for ACE-Value, and vice versa: $\theta = 0.3$ is the best threshold for the entity F-measure, while $\theta = 0.5$ is optimal for ACE-Value. This is highlighted in Table 2, where row "B-opt-F" contains the best results optimizing the entity F-measure (at $\theta = 0.3$), row "B-opt-AV" contains the best results optimizing ACE-Value (at $\theta = 0.5$), and the last line "Twin-model" contains the results of the proposed twin-model. It is clear from Table 2 that thresholding cannot be used to optimize the entity F-measure and ACE-Value simultaneously. A sub-optimal threshold could be detrimental to an unweighted metric such as the entity F-measure. The proposed twin model eliminates the need for thresholding, a benefit of using the principled creation model. In practice, the optimal threshold is a free parameter that has to be tuned every time when a task, dataset and model changes. Thus the proposed twin model is more portable when a task or dataset changes.

System	F-measure	ACE-Value
B-opt-F	84.7	87.5
B-opt-AV	81.1	88.0
Twin-model	**84.8**	**88.4**

Table 2: Comparison between the thresholding baseline and the twin model: optimal threshold depends on performance metric. The proposed twin-model outperforms the baseline without tuning the free parameter.

5 Related Work

Some earlier work (Lappin and Leass, 1994; Kennedy and Boguraev, 1996) use heuristic to determine whether a phrase is anaphoric or not. Bean and Riloff (1999) extracts rules from non-anaphoric noun phrases and noun phrases patterns, which are then applied to test data to identify *existential* noun phrases. It is intended as as pre-filtering step before a coreference resolution system is run. Ng and Cardie (2002a) trains a separate anaphoricity classifier in addition to a coreference model. The anaphoricity classifier is applied as a filter and only anaphoric mentions are later considered by the coreference model. Ng (2004) studies what is the best way to make use of anaphoricity information and concludes that the constrained-based and globally-optimized approach works the best. Poesio et al. (2004) contains a good summary of recent research work on discourse new or anaphoricity. Luo et al. (2004) uses a *start* model to determine whether a mention is the first one in a coreference chain, but it is computed ad hoc without training. Nicolae and Nicolae (2006) constructs a graph where mentions are nodes and an edge represents the likelihood two mentions are in an entity, and then a graph-cut algorithm is employed to produce final coreference results.

We take the view that determining whether an anaphor is coreferential with any candidate antecedent is part of the coreference process. But we do recognize that the disparity between the two types of events: while a coreferential relationship can be resolved by examining the local context of the anaphor and its antecedent, it is necessary to compare the anaphor with all the preceding candidates before it can be declared that it is not coreferential with any. Thus, a *creation* component $P_c(\cdot|E_t, m_t)$ is needed to model the second type of events. A problem arising from the adoption of the creation model is that it is very expensive to have a conditional model depending on all preceding entities E_t. To solve this problem, we adopt the MaxEnt model and impose some reasonable constraints on the feature functions, which makes it possible to synthesize features in the creation model from those of the link model. The twin model components are intimately trained and used simultaneously in our coreference system.

6 Conclusions

A twin-model is proposed for coreference resolution: one *link* component computes how likely a mention is coreferential with a candidate entity; the other component, called *creation* model, computes the probability that a mention is not coreferential with any candidate entity. Log linear or MaxEnt approach is adopted for building the two components. The twin components are trained and used simultaneously in our coreference system.

The creation model depends on all preceding entities and is often expensive to compute. We impose some reasonable constraints on feature functions which makes it feasible to compute efficiently the features in the creation model from a subset of link feature values. We test the proposed twin-model on the ACE 2005 data and the proposed model outperforms a thresholding baseline. Moreover, it is observed that the optimal threshold in the baseline depends on performance metric, while the proposed model eliminates the need of tuning the optimal threshold.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency under contract No. HR0011-06-2-0001. 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.

I would like to thank Salim Roukos for helping to improve the writing of the paper. Suggestions and comments from three anonymous reviewers are also gratefully acknowledged.

References

David L. Bean and Ellen Riloff. 1999. Corpus-based identification of non-anaphoric noun phrases. In *Proc. ACL*.

E.T. Bell. 1934. Exponential numbers. *Amer. Math. Monthly*, pages 411–419.

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.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proc. of HLT and EMNLP*, pages 97–104, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Pro. of the 40th ACL*.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of COLING-96 (16th International Conference on Computational Linguistics)*, Copenhagen,DK.

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

Xiaoqiang Luo and Imed Zitouni. 2005. Multi-lingual coreference resolution with syntactic features. In *Proc. of Human Language Technology (HLT)/Empirical Methods in Natural Language Processing (EMNLP)*.

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*.

Vincent Ng and Claire Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of COLING*.

Vincent Ng and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*, pages 104–111.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 151–158, Barcelona, Spain, July.

Cristina Nicolae and Gabriel Nicolae. 2006. BEST-CUT: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 275–283, Sydney, Australia, July. Association for Computational Linguistics.

NIST. 2005. ACE 2005 evaluation. www.nist.gov/speech/tests/ace/ace05/index.htm.

M. Poesio, O. Uryupina, R. Vieira, M. Alexandrov-Kabadjov, and R. Goulart. 2004. Discourse-new detectors for definite description resolution: A survey and a preliminary proposal. In *ACL 2004: Workshop on Reference Resolution and its Applications*, pages 47–54, Barcelona, Spain, July.

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.

Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proc. of the* 41^{st} *ACL*.

First-Order Probabilistic Models for Coreference Resolution

Aron Culotta and **Michael Wick** and **Andrew McCallum**
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{culotta,mwick,mccallum}@cs.umass.edu

Abstract

Traditional noun phrase coreference resolution systems represent features only of pairs of noun phrases. In this paper, we propose a machine learning method that enables features over *sets* of noun phrases, resulting in a first-order probabilistic model for coreference. We outline a set of approximations that make this approach practical, and apply our method to the ACE coreference dataset, achieving a 45% error reduction over a comparable method that only considers features of pairs of noun phrases. This result demonstrates an example of how a first-order logic representation can be incorporated into a probabilistic model and scaled efficiently.

1 Introduction

Noun phrase coreference resolution is the problem of clustering noun phrases into anaphoric sets. A standard machine learning approach is to perform a set of independent binary classifications of the form "*Is mention a coreferent with mention b?*"

This approach of decomposing the problem into pairwise decisions presents at least two related difficulties. First, it is not clear how best to convert the set of pairwise classifications into a disjoint clustering of noun phrases. The problem stems from the transitivity constraints of coreference: If a and b are coreferent, and b and c are coreferent, then a and c must be coreferent.

This problem has recently been addressed by a number of researchers. A simple approach is to perform the transitive closure of the pairwise decisions. However, as shown in recent work (McCallum and Wellner, 2003; Singla and Domingos, 2005), better performance can be obtained by performing *relational inference* to directly consider the dependence among a set of predictions. For example, McCallum and Wellner (2005) apply a graph partitioning algorithm on a weighted, undirected graph in which vertices are noun phrases and edges are weighted by the pairwise score between noun phrases.

A second and less studied difficulty is that the pairwise decomposition restricts the feature set to evidence about *pairs* of noun phrases only. This restriction can be detrimental if there exist features of *sets* of noun phrases that cannot be captured by a combination of pairwise features. As a simple example, consider prohibiting coreferent sets that consist only of pronouns. That is, we would like to require that there be at least one antecedent for a set of pronouns. The pairwise decomposition does not make it possible to capture this constraint.

In general, we would like to construct arbitrary features over a cluster of noun phrases using the full expressivity of first-order logic. Enabling this sort of flexible representation within a statistical model has been the subject of a long line of research on *first-order probabilistic models* (Gaifman, 1964; Halpern, 1990; Paskin, 2002; Poole, 2003; Richardson and Domingos, 2006).

Conceptually, a first-order probabilistic model can be described quite compactly. A configuration of the world is represented by a set of predi-

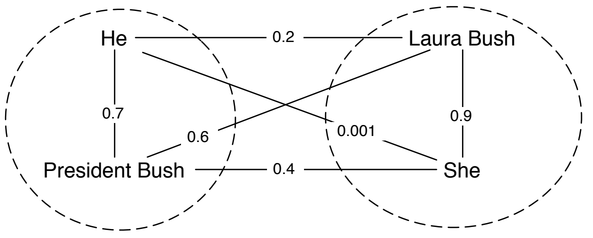


Figure 1: An example noun coreference graph in which vertices are noun phrases and edge weights are proportional to the probability that the two nouns are coreferent. Partitioning such a graph into disjoint clusters corresponds to performing coreference resolution on the noun phrases.

cates, each of which has an associated real-valued parameter. The likelihood of each configuration of the world is proportional to a combination of these weighted predicates. In practice, however, enumerating all possible configurations, or even all the predicates of one configuration, can result in intractable combinatorial growth (de Salvo Braz et al., 2005; Culotta and McCallum, 2006).

In this paper, we present a practical method to perform training and inference in first-order models of coreference. We empirically validate our approach on the ACE coreference dataset, showing that the first-order features can lead to an 45% error reduction.

2 Pairwise Model

In this section we briefly review the standard pairwise coreference model. Given a pair of noun phrases $x_{ij} = \{x_i, x_j\}$, let the binary random variable y_{ij} be 1 if x_i and x_j are coreferent. Let $F = \{f_k(x_{ij}, y)\}$ be a set of features over x_{ij}. For example, $f_k(x_{ij}, y)$ may indicate whether x_i and x_j have the same gender or number. Each feature f_k has an associated real-valued parameter λ_k. The pairwise model is

$$p(y_{ij}|x_{ij}) = \frac{1}{Z_{x_{ij}}} \exp \sum_k \lambda_k f_k(x_{ij}, y_{ij})$$

where $Z_{x_{ij}}$ is a normalizer that sums over the two settings of y_{ij}.

This is a maximum-entropy classifier (i.e. logistic regression) in which $p(y_{ij}|x_{ij})$ is the probability that x_i and x_j are coreferent. To estimate $\Lambda = \{\lambda_k\}$ from labeled training data, we perform gradient ascent to maximize the log-likelihood of the labeled data.

Two critical decisions for this method are (1) how to sample the training data, and (2) how to combine the pairwise predictions at test time. Systems often perform better when these decisions complement each other.

Given a data set in which noun phrases have been manually clustered, the training data can be created by simply enumerating over each pair of noun phrases x_{ij}, where y_{ij} is true if x_i and x_j are in the same cluster. However, this approach generates a highly unbalanced training set, with negative examples outnumbering positive examples. Instead, Soon et al. (2001) propose the following sampling method: Scan the document from left to right. Compare each noun phrase x_i to each preceding noun phrase x_j, scanning from right to left. For each pair x_i, x_j, create a training instance $\langle x_{ij}, y_{ij} \rangle$, where y_{ij} is 1 if x_i and x_j are coreferent. The scan for x_j terminates when a positive example is constructed, or the beginning of the document is reached. This results in a training set that has been pruned of distant noun phrase pairs.

At testing time, we can construct an undirected, weighted graph in which vertices correspond to noun phrases and edge weights are proportional to $p(y_{ij}|x_{ij})$. The problem is then to partition the graph into clusters with high *intra-cluster* edge weights and low *inter-cluster* edge weights. An example of such a graph is shown in Figure 1.

Any partitioning method is applicable here; however, perhaps most common for coreference is to perform greedy clustering guided by the word order of the document to complement the sampling method described above (Soon et al., 2001). More precisely, scan the document from left-to-right, assigning each noun phrase x_i to the same cluster as the closest *preceding* noun phrase x_j for which $p(y_{ij}|x_{ij}) > \delta$, where δ is some classification threshold (typically 0.5). Note that this method contrasts with standard greedy agglomerative clustering, in which each noun phrase would be assigned to the *most probable* cluster according to $p(y_{ij}|x_{ij})$.

Choosing the closest preceding phrase is common because nearby phrases are a priori more likely to be coreferent.

We refer to the training and inference methods described in this section as the Pairwise Model.

3 First-Order Logic Model

We propose augmenting the Pairwise Model to enable classification decisions over *sets* of noun phrases.

Given a set of noun phrases $\mathbf{x}^j = \{x_i\}$, let the binary random variable y_j be 1 if *all* the noun phrases $x_i \in \mathbf{x}^j$ are coreferent. The features f_k and weights λ_k are defined as before, but now the features can represent arbitrary attributes over the entire set $\mathbf{x}^j$. This allows us to use the full flexibility of first-order logic to construct features about sets of nouns. The First-Order Logic Model is

$$p(y_j|\mathbf{x}^j) = \frac{1}{Z_{\mathbf{x}^j}} \exp \sum_k \lambda_k f_k(\mathbf{x}^j, y_j)$$

where $Z_{\mathbf{x}^j}$ is a normalizer that sums over the two settings of y_j.

Note that this model gives us the representational power of recently proposed Markov logic networks (Richardson and Domingos, 2006); that is, we can construct arbitrary formulae in first-order logic to characterize the noun coreference task, and can learn weights for instantiations of these formulae. However, naively *grounding* the corresponding Markov logic network results in a combinatorial explosion of variables. Below we outline methods to scale training and prediction with this representation.

As in the Pairwise Model, we must decide how to sample training examples and how to combine independent classifications at testing time. It is important to note that by moving to the First-Order Logic Model, the number of possible predictions has increased exponentially. In the Pairwise Model, the number of possible y variables is $O(|\mathbf{x}|^2)$, where $\mathbf{x}$ is the set of noun phrases. In the First-Order Logic Model, the number of possible y variables is $O(2^{|\mathbf{x}|})$: There is a y variable for each possible element of the powerset of $\mathbf{x}$. Of course, we do not enumerate this set; rather, we incrementally instantiate y variables as needed during prediction.

A simple method to generate training examples is to sample positive and negative cluster examples uniformly at random from the training data. Positive examples are generated by first sampling a true cluster, then sampling a subset of that cluster. Negative examples are generated by sampling two positive examples and merging them into the same cluster.

At testing time, we perform standard greedy agglomerative clustering, where the score for each merger is proportional to the probability of the newly formed clustering according to the model. Clustering terminates when there exists no additional merge that improves the probability of the clustering.

We refer to the system described in this section as First-Order Uniform.

4 Error-driven and Rank-based training of the First-Order Model

In this section we propose two enhancements to the training procedure for the First-Order Uniform model.

First, because each training example consists of a subset of noun phrases, the number of possible training examples we can generate is exponential in the number of noun phrases. We propose an error-driven sampling method that generates training examples from errors the model makes on the training data. The algorithm is as follows: Given initial parameters Λ, perform greedy agglomerative clustering on training document i until an incorrect cluster is formed. Update the parameter vector according to this mistake, then repeat for the next training document. This process is repeated for a fixed number of iterations.

Exactly how to update the parameter vector is addressed by the second enhancement. We propose modifying the optimization criterion of training to perform *ranking* rather than *classification* of clusters. Consider a training example cluster with a negative label, indicating that not all of the noun phrases it contains are coreferent. A classification training algorithm will "penalize" all the features associated with this cluster, since they correspond to a negative example. However, because there may exists subsets of the cluster that *are* coreferent, features representing these positive subsets may be unjustly penalized.

To address this problem, we propose constructing training examples consisting of one negative exam-

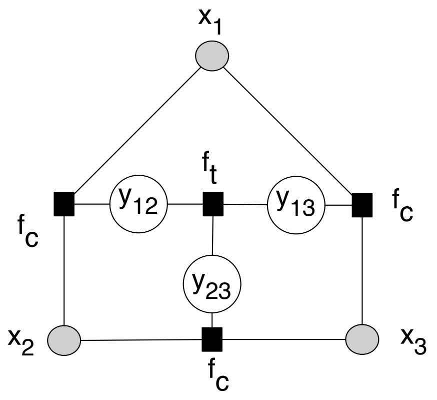


Figure 2: An example noun coreference factor graph for the Pairwise Model in which factors f_c model the coreference between two nouns, and f_t enforce the transitivity among related decisions. The number of y variables increases quadratically in the number of x variables.

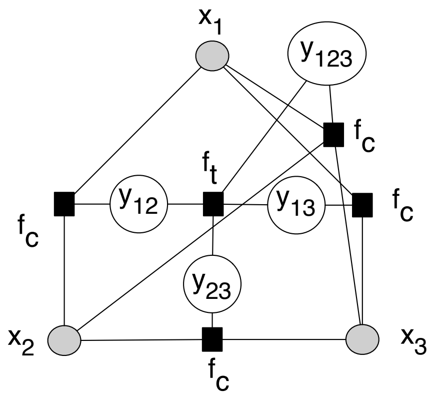


Figure 3: An example noun coreference factor graph for the First-Order Model in which factors f_c model the coreference between *sets* of nouns, and f_t enforce the transitivity among related decisions. Here, the additional node y_{123} indicates whether nouns $\{x_1, x_2, x_3\}$ are all coreferent. The number of y variables increases exponentially in the number of x variables.

ple and one "nearby" positive example. In particular, when agglomerative clustering incorrectly merges two clusters, we select the resulting cluster as the negative example, and select as the positive example a cluster that can be created by merging other existing clusters.[1] We then update the weight vector so that the positive example is assigned a higher score than the negative example. This approach allows the update to only penalize the *difference* between the two features of examples, thereby not penalizing features representing any overlapping coreferent clusters.

To implement this update, we use MIRA (Margin Infused Relaxed Algorithm), a relaxed, online maximum margin training algorithm (Crammer and Singer, 2003). It updates the parameter vector with two constraints: (1) the positive example must have a higher score by a given margin, and (2) the change to Λ should be minimal. This second constraint is to reduce fluctuations in Λ. Let $s^+(\Lambda, \mathbf{x}^j)$ be the unnormalized score for the positive example and $s^-(\Lambda, \mathbf{x}^k)$ be the unnormalized score of the negative example. Each update solves the following quadratic program:

$$\Lambda^{t+1} = \underset{\Lambda}{\operatorname{argmin}}\, ||\Lambda^t - \Lambda||^2$$
$$\text{s.t.}$$
$$s^+(\Lambda, \mathbf{x}^j) - s^-(\Lambda, \mathbf{x}^k) \geq 1$$

In this case, MIRA with a single constraint can be efficiently solved in one iteration of the Hildreth and D'Esopo method (Censor and Zenios, 1997). Additionally, we average the parameters calculated at each iteration to improve convergence.

We refer to the system described in this section as First-Order MIRA.

5 Probabilistic Interpretation

In this section, we describe the Pairwise and First-Order models in terms of the factor graphs they approximate.

For the Pairwise Model, a corresponding undirected graphical model can be defined as

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \prod_{y_{ij} \in \mathbf{y}} f_c(y_{ij}, x_{ij}) \prod_{y_{ij}, y_{jk} \in \mathbf{y}} f_t(y_{ij}, y_{j,k}, y_{ik}, x_{ij}, x_{jk}, x_{ik})$$

[1] Of the possible positive examples, we choose the one with the highest probability under the current model to guard against large fluctuations in parameter updates

where $Z_{\mathbf{x}}$ is the input-dependent normalizer and factor f_c parameterizes the pairwise noun phrase compatibility as $f_c(y_{ij}, x_{ij}) = \exp(\sum_k \lambda_k f_k(y_{ij}, x_{ij}))$. Factor f_t enforces the transitivity constraints by $f_t(\cdot) = -\infty$ if transitivity is not satisfied, 1 otherwise. This is similar to the model presented in McCallum and Wellner (2005). A factor graph for the Pairwise Model is presented in Figure 2 for three noun phrases.

For the First-Order model, an undirected graphical model can be defined as

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \prod_{y_j \in \mathbf{y}} f_c(y_j, \mathbf{x}^j) \prod_{y_j \in \mathbf{y}} f_t(y_j, \mathbf{x}^j)$$

where $Z_{\mathbf{x}}$ is the input-dependent normalizer and factor f_c parameterizes the cluster-wise noun phrase compatibility as $f_c(y_j, \mathbf{x}^j) = \exp(\sum_k \lambda_k f_k(y_j, x^j))$. Again, factor f_t enforces the transitivity constraints by $f_t(\cdot) = -\infty$ if transitivity is not satisfied, 1 otherwise. Here, transitivity is a bit more complicated, since it also requires that if $y_j = 1$, then for any subset $\mathbf{x}^k \subseteq \mathbf{x}^j$, $y_k = 1$. A factor graph for the First-Order Model is presented in Figure 3 for three noun phrases.

The methods described in Sections 2, 3 and 4 can be viewed as estimating the parameters of each factor f_c independently. This approach can therefore be viewed as a type of *piecewise approximation* of exact parameter estimation in these models (Sutton and McCallum, 2005). Here, each f_c is a "piece" of the model trained independently. These pieces are combined at prediction time using clustering algorithms to enforce transitivity. Sutton and McCallum (2005) show that such a piecewise approximation can be theoretically justified as minimizing an upper bound of the exact loss function.

6 Experiments

6.1 Data

We apply our approach to the noun coreference ACE 2004 data, containing 443 news documents with 28,135 noun phrases to be coreferenced. 336 documents are used for training, and the remainder for testing. All entity types are candidates for coreference (pronouns, named entities, and nominal entities). We use the true entity segmentation, and parse each sentence in the corpus using a phrase-structure grammar, as is common for this task.

6.2 Features

We follow Soon et al. (2001) and Ng and Cardie (2002) to generate most of our features for the Pairwise Model. These include:

- Match features - Check whether gender, number, head text, or entire phrase matches
- Mention type (pronoun, name, nominal)
- Aliases - Heuristically decide if one noun is the acronym of the other
- Apposition - Heuristically decide if one noun is in apposition to the other
- Relative Pronoun - Heuristically decide if one noun is a relative pronoun referring to the other.
- Wordnet features - Use Wordnet to decide if one noun is a hypernym, synonym, or antonym of another, or if they share a hypernym.
- Both speak - True if both contain an adjacent context word that is a synonym of "said." This is a domain-specific feature that helps for many newswire articles.
- Modifiers Match - for example, in the phrase "President Clinton", "President" is a modifier of "Clinton". This feature indicates if one noun is a modifier of the other, or they share a modifier.
- Substring - True if one noun is a substring of the other (e.g. "Egypt" and "Egyptian").

The First-Order Model includes the following features:

- Enumerate each pair of noun phrases and compute the features listed above. **All-X** is true if all pairs share a feature X, **Most-True-X** is true if the majority of pairs share a feature X, and **Most-False-X** is true if most of the pairs do not share feature X.

- Use the output of the Pairwise Model for each pair of nouns. **All-True** is true if all pairs are predicted to be coreferent, **Most-True** is true if most pairs are predicted to be coreferent, and **Most-False** is true if most pairs are predicted to not be coreferent. Additionally, **Max-True** is true if the maximum pairwise score is above threshold, and **Min-True** if the minimum pairwise score is above threshold.

- Cluster Size indicates the size of the cluster.

- Count how many phrases in the cluster are of each mention type (name, pronoun, nominal), number (singular/plural) and gender (male/female). The features **All-X** and **Most-True-X** indicate how frequent each feature is in the cluster. This feature can capture the soft constraint such that no cluster consists only of pronouns.

In addition to the listed features, we also include conjunctions of size 2, for example "Genders match AND numbers match".

6.3 Evaluation

We use the B^3 algorithm to evaluate the predicted coreferent clusters (Amit and Baldwin, 1998). B^3 is common in coreference evaluation and is similar to the precision and recall of coreferent links, except that systems are rewarded for singleton clusters. For each noun phrase x_i, let c_i be the number of mentions in x_i's predicted cluster that are in fact coreferent with x_i (including x_i itself). Precision for x_i is defined as c_i divided by the number of noun phrases in x_i's cluster. Recall for x_i is defined as the c_i divided by the number of mentions in the gold standard cluster for x_i. $F1$ is the harmonic mean of recall and precision.

6.4 Results

In addition to Pairwise, First-Order Uniform, and First-Order MIRA, we also compare against Pairwise MIRA, which differs from First-Order MIRA only by the fact that it is restricted to pairwise features.

Table 1 suggests both that first-order features and error-driven training can greatly improve performance. The First-Order Model outperforms the Pairwise Model in F1 measure for both standard training and error-driven training. We attribute some of this improvement to the capability of the First-Order model to capture features of entire clusters that may indicate some phrases are not coreferent. Also, we attribute the gains from error-driven training to the fact that training examples are generated based on errors made on the training data. (However, we should note that there are also small differences in the feature sets used for error-driven and standard training results.)

	F1	**Prec**	**Rec**
First-Order MIRA	**79.3**	86.7	73.2
Pairwise MIRA	72.5	92.0	59.8
First-Order Uniform	**69.2**	79.0	61.5
Pairwise	62.4	62.5	62.3

Table 1: B^3 results for ACE noun phrase coreference. FIRST-ORDER MIRA is our proposed model that takes advantage of first-order features of the data and is trained with error-driven and rank-based methods. We see that both the first-order features and the training enhancements improve performance consistently.

Error analysis indicates that often noun x_i is correctly not merged with a cluster $\mathbf{x}^j$ when $\mathbf{x}^j$ has a strong internal coherence. For example, if all 5 mentions of *France* in a document are string identical, then the system will be extremely cautious of merging a noun that is not equivalent to *France* into $\mathbf{x}^j$, since this will turn off the "All-String-Match" feature for cluster $\mathbf{x}^j$.

To our knowledge, the best results on this dataset were obtained by the meta-classification scheme of Ng (2005). Although our train-test splits may differ slightly, the best B-Cubed F1 score reported in Ng (2005) is 69.3%, which is considerably lower than the 79.3% obtained with our method. Also note that the Pairwise baseline obtains results similar to those in Ng and Cardie (2002).

7 Related Work

There has been a recent interest in training methods that enable the use of first-order features (Paskin, 2002; Daumé III and Marcu, 2005b; Richardson and Domingos, 2006). Perhaps the most related is "learning as search optimization" (LASO) (Daumé III and Marcu, 2005b; Daumé III and Marcu, 2005a). Like the current paper, LASO is also an error-driven training method that integrates prediction and training. However, whereas we explicitly use a ranking-based loss function, LASO uses a binary classification loss function that labels each candidate structure as *correct* or *incorrect*. Thus, each LASO training example contains *all* candidate predictions, whereas our training examples contain only the highest scoring incorrect prediction and the highest scoring correct prediction. Our experiments show the advantages of this ranking-based loss function. Additionally, we provide an empirical study to quantify the effects of different example generation and loss function decisions.

Collins and Roark (2004) present an incremental perceptron algorithm for parsing that uses "early update" to update the parameters when an error is encountered. Our method uses a similar "early update" in that training examples are only generated for the *first* mistake made during prediction. However, they do not investigate rank-based loss functions.

Others have attempted to train global scoring functions using Gibbs sampling (Finkel et al., 2005), message propagation, (Bunescu and Mooney, 2004; Sutton and McCallum, 2004), and integer linear programming (Roth and Yih, 2004). The main distinctions of our approach are that it is simple to implement, not computationally intensive, and adaptable to arbitrary loss functions.

There have been a number of machine learning approaches to coreference resolution, traditionally factored into classification decisions over pairs of nouns (Soon et al., 2001; Ng and Cardie, 2002). Nicolae and Nicolae (2006) combine pairwise classification with graph-cut algorithms. Luo et al. (2004) do enable features between mention-cluster pairs, but do not perform the error-driven and ranking enhancements proposed in our work. Denis and Baldridge (2007) use a ranking loss function for pronoun coreference; however the examples are still pairs of pronouns, and the example generation is not error driven. Ng (2005) learns a meta-classifier to choose the best prediction from the output of several coreference systems. While in theory a meta-classifier can flexibly represent features, they do not explore features using the full flexibility of first-order logic. Also, their method is neither error-driven nor rank-based.

McCallum and Wellner (2003) use a conditional random field that factors into a product of pairwise decisions about pairs of nouns. These pairwise decisions are made collectively using relational inference; however, as pointed out in Milch et al. (2004), this model has limited representational power since it does not capture features of *entities*, only of pairs of mention. Milch et al. (2005) address these issues by constructing a generative probabilistic model, where noun clusters are sampled from a generative process. Our current work has similar representational flexibility as Milch et al. (2005) but is discriminatively trained.

8 Conclusions and Future Work

We have presented learning and inference procedures for coreference models using first-order features. By relying on sampling methods at training time and approximate inference methods at testing time, this approach can be made scalable. This results in a coreference model that can capture features over *sets* of noun phrases, rather than simply *pairs* of noun phrases.

This is an example of a model with extremely flexible representational power, but for which exact inference is intractable. The simple approximations we have described here have enabled this more flexible model to outperform a model that is simplified for tractability.

A short-term extension would be to consider features over entire *clusterings*, such as the number of clusters. This could be incorporated in a ranking scheme, as in Ng (2005).

Future work will extend our approach to a wider variety of tasks. The model we have described here is specific to clustering tasks; however a similar formulation could be used to approach a number of language processing tasks, such as parsing and relation extraction. These tasks could benefit from first-order features, and the present work can guide the approximations required in those domains.

Additionally, we are investigating more sophisticated inference algorithms that will reduce the greediness of the search procedures described here.

Acknowledgments

We thank Robert Hall for helpful contributions. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract #NBCHD030010, in part by U.S. Government contract #NBCH040171 through a subcontract with BBNT Solutions LLC, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, in part by Microsoft Live Labs, and in part by the Defense Advanced Research Projects Agency (DARPA) under contract #HR0011-06-C-0023. 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

B. Amit and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Seventh Message Understanding Conference (MUC7)*.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *ACL*.

Y. Censor and S.A. Zenios. 1997. *Parallel optimization : theory, algorithms, and applications*. Oxford University Press.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL*.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*, 3:951–991.

Aron Culotta and Andrew McCallum. 2006. Tractable learning and inference with high-order representations. In *ICML Workshop on Open Problems in Statistical Relational Learning*, Pittsburgh, PA.

Hal Daumé III and Daniel Marcu. 2005a. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *HLT/EMNLP*, Vancouver, Canada.

Hal Daumé III and Daniel Marcu. 2005b. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*, Bonn, Germany.

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. 2005. Lifted first-order probabilistic inference. In *IJCAI*, pages 1319–1325.

Pascal Denis and Jason Baldridge. 2007. A ranking approach to pronoun resolution. In *IJCAI*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370.

H. Gaifman. 1964. Concerning measures in first order calculi. *Israel J. Math*, 2:1–18.

J. Y. Halpern. 1990. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *ACL*, page 135.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*.

Andrew McCallum and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *NIPS17*. MIT Press, Cambridge, MA.

Brian Milch, Bhaskara Marthi, and Stuart Russell. 2004. BLOG: Relational modeling with unknown objects. In *ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. 2005. BLOG: Probabilistic models with unknown objects. In *IJCAI*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *ACL*.

Cristina Nicolae and Gabriel Nicolae. 2006. Bestcut: A graph algorithm for coreference resolution. In *EMNLP*, pages 275–283, Sydney, Australia, July. Association for Computational Linguistics.

Mark A. Paskin. 2002. Maximum entropy probabilistic logic. Technical Report UCB/CSD-01-1161, University of California, Berkeley.

D. Poole. 2003. First-order probabilistic inference. In *IJCAI*, pages 985–991, Acapulco, Mexico. Morgan Kaufman.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *The 8th Conference on Compuational Natural Language Learning*, May.

Parag Singla and Pedro Domingos. 2005. Discriminative training of markov logic networks. In *AAAI*, Pittsburgh, PA.

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

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July.

Charles Sutton and Andrew McCallum. 2005. Piecewise training of undirected models. In *21st Conference on Uncertainty in Artificial Intelligence*.

Information Retrieval On Empty Fields

Victor Lavrenko, Xing Yi and James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610, USA
{lavrenko,yixing,allan}@cs.umass.edu

Abstract

We explore the problem of retrieving semi-structured documents from a real-world collection using a structured query. We formally develop Structured Relevance Models (SRM), a retrieval model that is based on the idea that plausible values for a given field could be inferred from the context provided by the other fields in the record. We then carry out a set of experiments using a snapshot of the National Science Digital Library (NSDL) repository, and queries that only mention fields missing from the test data. For such queries, typical field matching would retrieve no documents at all. In contrast, the SRM approach achieves a mean average precision of over twenty percent.

1 Introduction

This study investigates information retrieval on semi-structured information, where documents consist of several textual fields that can be queried independently. If documents contained *subject* and *author* fields, for example, we would expect to see queries looking for documents about *theory of relativity* by the author *Einstein*.

This setting suggests exploring the issue of inexact match—is *special theory of relativity* relevant?—that has been explored elsewhere (Cohen, 2000). Our interest is in an extreme case of that problem, where the content of a field is not corrupted or incorrect, but is actually absent. We wish to find relevant information in response to a query such as the one above even if a relevant document is completely missing the *subject* and *author* fields.

Our research is motivated by the challenges we encountered in working with the National Science Digital Library (NSDL) collection.[1] Each item in the collection is a scientific resource, such as a research paper, an educational video, or perhaps an entire website. In addition to its main content, each resource is annotated with *metadata*, which provides information such as the author or creator of the resource, its subject area, format (text/image/video) and intended audience – in all over 90 distinct fields (though some are very related). Making use of such extensive metadata in a digital library paves the way for constructing highly-focused models of the user's information need. These models have the potential to dramatically improve the user experience in targeted applications, such as the NSDL portals. To illustrate this point, suppose that we are running an educational portal targeted at elementary school teachers, and some user requests teaching aids for an introductory class on gravity. An intelligent search system would be able to translate the request into a structured query that might look something like: *subject='gravity' AND audience='grades 1-4' AND format='image,video' AND rights='free-for-academic-use'*. Such a query can be efficiently answered by a relational database system.

Unfortunately, using a relational engine to query a semi-structured collection similar to NSDL will run into a number of obstacles. The simplest problem is

[1]http://www.nsdl.org

that natural language fields are filled inconsistently: e.g., the *audience* field contains values such as *K-4, K-6, second grade*, and *learner*, all of which are clearly semantically related.

A larger problem, and the one we focus on in this study, is that of missing fields. For example 24% of the items in the NSDL collection have no subject field, 30% are missing the author information, and over 96% mention no target audience (reading level). This means that a relational query for elementary school material will consider at most 4% of all potentially relevant resources in the NSDL collection.[2]

The goal of our work is to introduce a retrieval model that will be capable of answering complex structured queries over a semi-structured collection with corrupt and missing field values. This study focuses on the latter problem, an extreme version of the former. Our approach is to use a generative model to compute how plausible a word would appear in a record's empty field given the context provided by the other fields in the record.

The remainder of this paper is organized as follows. We survey previous attempts at handling semi-structured data in section 2. Section 3 will provide the details of our approach, starting with a high-level view, then providing a mathematical framework, and concluding with implementation details. Section 4 will present an extensive evaluation of our model on the large set of queries over the NSDL collection. We will summarize our results and suggest directions for future research in Section 5.

2 Related work

The issue of missing field values is addressed in a number of recent publications straddling the areas of relational databases and machine learning. In most cases, researchers introduce a statistical model for predicting the value of a missing attribute or relation, based on observed values. Friedman et al (1999) introduce a technique called Probabilistic Relational Models (PRM) for automatically learning the structure of dependencies in a relational database. Taskar et al (2001) demonstrate how PRM can be used to predict the category of a given research paper and show that categorization accuracy can be substantially improved by leveraging the relational structure of the data. Heckerman et al (2004) introduce the Probabilistic Entity Relationship (PER) model as an extension of PRM that treats relations between entities as objects. Neville at al (2003) discuss predicting binary labels in relational data using Relational Probabilistic Trees (RPT). Using this method they successfully predict whether a movie was a box office hit based on other movies that share some of the properties (actors, directors, producers) with the movie in question.

Our work differs from most of these approaches in that we work with free-text fields, whereas database researchers typically deal with closed-vocabulary values, which exhibit neither the synonymy nor the polysemy inherent in natural language expressions. In addition, the goal of our work is different: we aim for accurate *ranking* of records by their relevance to the user's query, whereas database research has typically focused on *predicting* the missing value.

Our work is related to a number of existing approaches to semi-structured text search. Desai et al (1987) followed by Macleod (1991) proposed using the standard relational approach to searching unstructured texts. The lack of an explicit ranking function in their approaches was partially addressed by Blair (1988). Fuhr (1993) proposed the use of Probabilistic Relational Algebra (PRA) over the weights of individual term matches. Vasanthukumar et al (1996) developed a relational implementation of the inference network retrieval model. A similar approach was taken by de Vries and Wilschut (1999), who managed to improve the efficiency of the approach. De Fazio et al (1995) integrated IR and RDBMS technology using an approached called cooperative indexing. Cohen (2000) describes WHIRL – a language that allows efficient inexact matching of textual fields within SQL statements. A number of relevant works are also published in the proceedings of the *INEX* workshop.[3]

The main difference between these endeavors and our work is that we are explicitly focusing on the cases where parts of the structured data are missing

[2]Some of the NSDL metadata fields overlap substantially in meaning, so it might be argued that the overlapping fields will cover the collection better. Under the broadest possible interpretation of field meanings, more than 7% of the documents still contain no subject and 95% still contain no audience field.

[3]http://inex.is.informatik.uni-duisburg.de/index.html

or mis-labeled.

3 Structured Relevance Model

In this section we will provide a detailed description of our approach to searching semi-structured data. Before diving into the details of our model, we want to clearly state the challenge we intend to address with our system.

3.1 Task: finding relevant records

The aim of our system is to identify a set of records relevant to a structured query provided by the user. We assume the query specifies a set of keywords for each field of interest to the user, for example *Q: subject='physics,gravity' AND audience='grades 1-4'*[4]. Each record in the database is a set of natural-language descriptions for each field. A record is considered relevant if it *could plausibly* be annotated with the query fields. For example, a record clearly aimed at elementary school students would be considered relevant to *Q* even if it does not contain *'grades 1-4'* in its description of the target audience. In fact, our experiments will specifically focus on finding relevant records that contain no direct match to the specified query fields, explicitly targeting the problem of missing data and inconsistent schemata.

This task is not a typical IR task because the fielded structure of the query is a critical aspect of the processing, not one that is largely ignored in favor of pure content based retrieval. On the other hand, the approach used is different from most DB work because cross-field dependencies are a key component of the technique. In addition, the task is unusual for both communities because it considers an unusual case where the fields in the query do not occur at all in the documents being searched.

3.2 Overview of the approach

Our approach is based on the idea that plausible values for a given field could be inferred from the context provided by the other fields in the record. For instance, a resource titled *'Transductive SVMs'* and containing highly technical language in its description is unlikely to be aimed at elementary-school students. In the following section we will describe a statistical model that will allow us to guess the values of un-observed fields. At the intuitive level, the model takes advantage of the fact that records similar in one respect will often be similar in others. For example, if two resources share the same author and have similar titles, they are likely to be aimed at the same audience. Formally, our model is based on the *generative* paradigm. We will describe a probabilistic process that could be viewed, hypothetically, as the source of every record in our collection. We will assume that the query provided by our user is also a sample from this generative process, albeit a very short one. We will use the observed query fields (e.g. *audience* and *subject*) to estimate the likely values for other fields, which would be *plausible* in the context of the observed subject and audience. The distributions over plausible values will be called *relevance models*, since they are intended to mimic the kind of record that might be relevant to the observed query. Finally, all records in the database will be ranked by their information-theoretic similarity to these relevance models.

3.3 Definitions

We start with a set of definitions that will be used through the remainder of this paper. Let C be a collection of semi-structured records. Each record $\mathbf{w}$ consists of a set of fields $\mathbf{w}_1 \ldots \mathbf{w}_m$. Each field $\mathbf{w}_i$ is a sequence of discrete variables (words) $\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,n_i}$, taking values in the field vocabulary $\mathcal{V}_i$.[5] When a record contains no information for the i'th field, we assume $n_i{=}0$ for that record. A user's query $\mathbf{q}$ takes the same representation as a record in the database: $\mathbf{q}{=}\{\mathbf{q}_{i,j}{\in}\mathcal{V}_i : i{=}1..m, j = 1..n_i\}$. We will use $\mathbf{p}_i$ to denote a *language model* over $\mathcal{V}_i$, i.e. a set of probabilities $\mathbf{p}_i(v){\in}[0,1]$, one for each word v, obeying the constraint $\Sigma_v \mathbf{p}_i(v) = 1$. The set of all possible language models over $\mathcal{V}_i$ will be denoted as the probability simplex $\mathbb{P}_i$. We define $\pi : \mathbb{P}_1 {\times} \cdots {\times} \mathbb{P}_m {\rightarrow} [0,1]$ to be a discrete measure function that assigns a probability mass $\pi(\mathbf{p}_1 \ldots \mathbf{p}_m)$ to a set of m language models, one for each of the m fields present in our collection.

[4]For this paper we will focus on simple conjunctive queries. Extending our model to more complex queries is reserved for future research.

[5]We allow each field to have its own vocabulary $\mathcal{V}_i$, since we generally do not expect author names to occur in the audience field, etc. We also allow $\mathcal{V}_i$ to share same words.

3.4 Generative Model

We will now present a generative process that will be viewed as a hypothetical source that produced every record in the collection C. We stress that this process is purely hypothetical; its only purpose is to model the kinds of dependencies that are necessary to achieve effective ranking of records in response to the user's query. We assume that each record $\mathbf{w}$ in the database is generated in the following manner:

1. Pick m distributions $\mathbf{p}_1 \ldots \mathbf{p}_m$ according to π
2. For each field $i = 1 \ldots m$:
   (a) Pick the length n_i of the $i'th$ field of $\mathbf{w}$
   (b) Draw i.i.d. words $\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,n_i}$ from $\mathbf{p}_i$

Under this process, the probability of observing a record $\{\mathbf{w}_{i,j} : i{=}1..m, j{=}1..n_i\}$ is given by the following expression:

$$\int_{\mathbb{P}_1 \ldots \mathbb{P}_m} \left[\prod_{i=1}^{m} \prod_{j=1}^{n_i} \mathbf{p}_i(\mathbf{w}_{i,j}) \right] \pi(\mathbf{p}_1 \ldots \mathbf{p}_m) d\mathbf{p}_1 \ldots d\mathbf{p}_m \quad (1)$$

3.4.1 A generative measure function

The generative measure function π plays a critical part in equation (1): it specifies the likelihood of using different combinations of language models in the process of generating $\mathbf{w}$. We use a non-parametric estimate for π, which relies directly on the combinations of language models that are observed in the training part of the collection. Each training record $\mathbf{w}_1 \ldots \mathbf{w}_m$ corresponds to a unique combination of language models $\mathbf{p}_1^{\mathbf{w}} \ldots \mathbf{p}_m^{\mathbf{w}}$ defined by the following equation:

$$\mathbf{p}_i^{\mathbf{w}}(v) = \frac{\#(v, \mathbf{w}_i) + \mu_i c_v}{n_i + \mu_i} \quad (2)$$

Here $\#(v, \mathbf{w}_i)$ represents the number of times the word v was observed in the i'th field of $\mathbf{w}$, n_i is the length of the i'th field, and c_v is the relative frequency of v in the entire collection. Meta-parameters μ_i allow us to control the amount of smoothing applied to language models of different fields; their values are set empirically on a held-out portion of the data.

We define $\pi(\mathbf{p}_1 \ldots \mathbf{p}_m)$ to have mass $\frac{1}{N}$ when its argument $\mathbf{p}_1 \ldots \mathbf{p}_m$ corresponds to one of the N records $\mathbf{w}$ in the training part C_t of our collection, and zero otherwise:

$$\pi(\mathbf{p}_1 \ldots \mathbf{p}_m) = \frac{1}{N} \sum_{\mathbf{w} \in C_t} \prod_{i=1}^{m} 1_{\mathbf{p}_i = \mathbf{p}_i^{\mathbf{w}}} \quad (3)$$

Here $\mathbf{p}_i^{\mathbf{w}}$ is the language model associated with the training record $\mathbf{w}$ (equation 2), and 1_x is the Boolean indicator function that returns 1 when its predicate x is true and zero when it is false.

3.4.2 Assumptions and limitations of the model

The generative model described in the previous section treats each field in the record as a *bag* of words with no particular order. This representation is often associated with the assumption of *word independence*. We would like to stress that our model does not assume word independence, on the contrary, it allows for strong *un-ordered* dependencies among the words – both within a field, and across different fields within a record. To illustrate this point, suppose we let $\mu_i \rightarrow 0$ in equation (2) to reduce the effects of smoothing. Now consider the probability of observing the word *'elementary'* in the audience field together with the word *'differential'* in the title (equation 1). It is easy to verify that the probability will be non-zero only if some training record $\mathbf{w}$ actually contained these words in their respective fields – an unlikely event. On the other hand, the probability of *'elementary'* and *'differential'* co-occurring in the same title might be considerably higher.

While our model does not assume word independence, it does ignore the relative ordering of the words in each field. Consequently, the model will fail whenever the order of words, or their proximity within a field carries a semantic meaning. Finally, our generative model does not capture dependencies across different records in the collection, each record is drawn independently according to equation (1).

3.5 Using the model for retrieval

In this section we will describe how the generative model described above can be used to find database records relevant to the structured query provided by the user. We are given a structured query $\mathbf{q}$, and a collection of records, partitioned into the training portion C_t and the testing portion C_e. We will use the training records to estimate a set of *relevance*

	records covered	average length	unique words
title	655,673 (99%)	7	102,772
description	514,092 (78%)	38	189,136
subject	504,054 (77%)	12	37,385
content	91,779 (14%)	743	575,958
audience	22,963 (3.5%)	4	119

Table 1: Summary statistics for the five NSDL fields used in our retrieval experiments.

models $R_1 \ldots R_m$, intended to reflect the user's information need. We will then rank testing records by their divergence from these relevance models. A relevance $R_i(v)$ specifies how plausible it is that word v would occur in the i'th field of a record, given that the record contains a perfect match to the query fields $\mathbf{q}_1 \ldots \mathbf{q}_m$:

$$R_i(v) = \frac{P(\mathbf{q}_1 \ldots v \circ \mathbf{q}_i \ldots \mathbf{q}_m)}{P(\mathbf{q}_1 \ldots \mathbf{q}_i \ldots \mathbf{q}_m)} \quad (4)$$

We use $v \circ \mathbf{q}_i$ to denote appending word v to the string $\mathbf{q}_i$. Both the numerator and the denominator are computed using equation (1). Once we have computed relevance models R_i for each of the m fields, we can rank testing records $\mathbf{w}'$ by their similarity to these relevance models. As a similarity measure we use weighted cross-entropy, which is an extension of the ranking formula originally proposed by (Lafferty and Zhai, 2001):

$$H(R_{1..m}; \mathbf{w}_{1..m}) = \sum_{i=1}^{m} \alpha_i \sum_{v \in \mathcal{V}_i} R_i(v) \log \mathbf{p}_i^{\mathbf{w}}(v) \quad (5)$$

The outer summation goes over every field of interest, while the inner extends over all the words in the vocabulary of the i'th field. R_i are computed according to equation (4), while $\mathbf{p}_i^{\mathbf{w}}$ are estimated from equation (2). Meta-parameters α_i allow us to vary the importance of different fields in the final ranking; the values are selected on a held-out portion of the data.

4 Experiments

4.1 Dataset and queries

We tested the performance of our model on a January 2005 snapshot of the National Science Digital Library repository. The snapshot contains a total of 656,992 records, spanning 92 distinct (though sometimes related) fields. [6]Only 7 of these fields are present in every record, and half the fields are present in less than 1% of the records. An average record contains only 17 of the 92 fields. Our experiments focus on a subset of 5 fields (*title*, *description*, *subject*, *content* and *audience*). These fields were selected for two reasons: (i) they occur frequently enough to allow a meaningful evaluation and (ii) they seem plausible to be included in a potential query.[7] Of these fields, *title* represents the title of the resource, *description* is a very brief abstract, *content* is a more detailed description (but not the full content) of the resource, *subject* is a library-like classification of the topic covered by the resource, and *audience* reflects the target reading level (e.g. *elementary school* or *post-graduate*). Summary statistics for these fields are provided in Table 1.

The dataset was randomly split into three subsets: the **training** set, which comprised 50% of the records and was used for estimating the relevance models as described in section 3.5; the **held-out** set, which comprised 25% of the data and was used to tune the smoothing parameters μ_i and the bandwidth parameters α_i; and the **evaluation** set, which contained 25% of the records and was used to evaluate the performance of the tuned model[8].

Our experiments are based on a set of 127 automatically generated queries. We randomly split the queries into two groups, 64 for training and 63 for evaluation. The queries were constructed by combining two randomly picked *subject* words with two *audience* words, and then discarding any combination that had less than 10 exact matches in any of the three subsets of our collection. This procedure yields queries such as Q_{91}=$\{$*subject:'artificial intelligence' AND audience='researchers'*$\}$, or Q_{101}=$\{$*subject:'philosophy' AND audience='high school'*$\}$.

4.2 Evaluation paradigm

We evaluate our model by its ability to find "relevant" records in the face of missing values. We de-

[6]As of May 2006, the NSDL contains over 1.5 million documents.

[7]The most frequent NSDL fields (*id, icon, url, link* and 4 *brand* fields) seem unlikely to be used in user queries.

[8]In real use, typical pseudo relevance feedback scheme can be followed: retrieve top-k documents to build relevance models then perform IR again on the same whole collection

fine a record $\mathbf{w}$ to be relevant to the user's query $\mathbf{q}$ if every keyword in $\mathbf{q}$ is found in the corresponding field of $\mathbf{w}$. For example, in order to be relevant to Q_{101} a record must contain the word *'philosophy'* in the subject field and words *'high'* and *'school'* in the audience field. If either of the keywords is missing, the record is considered non-relevant.[9]

When the testing records are fully observable, achieving perfect retrieval accuracy is trivial: we simply return all records that match all query keywords in the subject and audience fields. As we stated earlier, our main interest concerns the scenario when parts of the testing data are missing. We are going to simulate this scenario in a rather extreme manner by *completely* removing the *subject* and *audience* fields from all testing records. This means that a straightforward approach – matching query fields against record fields – will yield no relevant results. Our approach will rank testing records by comparing their *title, description* and *content* fields against the query-based relevance models, as discussed in section 3.5.

We will use the standard rank-based evaluation metrics: *precision* and *recall*. Let N_R be the total number of records relevant to a given query, suppose that the first K records in our ranking contain N_K relevant ones. Precision at rank K is defined as $\frac{N_K}{K}$ and recall is defined as $\frac{N_K}{N_R}$. Average precision is defined as the mean precision over all ranks where relevant items occur. R-precision is defined as precision at rank $K{=}N_R$.

4.3 Baseline systems

Our experiments will compare the ranking performance of the following retrieval systems:

cLM is a *cheating* version of un-structured text search using a state-of-the-art language-modeling approach (Ponte and Croft, 1998). We disregard the structure, take all query keywords and run them against a *concatenation* of all fields in the testing records. This is a "cheating" baseline, since the concatenation includes the *audience* and *subject* fields, which are supposed to be missing from the testing records. We use Dirichlet smoothing (Lafferty and Zhai, 2001), with parameters optimized on the training data. This baseline mimics the core search capability currently available on the NSDL website.

bLM is a combination of SQL-like structured matching and unstructured search with query expansion. We take all training records that contain an exact match to our query and select 10 highly-weighted words from the *title*, *description*, and *content* fields of these records. We run the resulting 30 words as a language modeling query against the concatenation of *title*, *description*, and *content* fields in the testing records. This is a non-cheating baseline.

bMatch is a structured extension of bLM. As in bLM, we pick training records that contain an exact match to the query fields. Then we match 10 highly-weighted *title* words, against the *title* field of testing records, do the same for the *description* and *content* fields, and merge the three resulting ranked lists. This is a non-cheating baseline that is similar to our model (SRM). The main difference is that this approach uses exact matching to select the training records, whereas SRM leverages a best-match language modeling algorithm.

SRM is the Structured Relevance Model, as described in section 3.5. For reasons of both effectiveness and efficiency, we firstly run the original query to retrieve top-500 records, then use these records to build SRMs. When calculating the cross entropy(equ. 5), for each field we only include the top-100 words which will appear in that field with the largest probabilities.

Note that our baselines do not include a standard SQL approach directly on testing records. Such an approach would have perfect performance in a "cheating" scenario with observable *subject* and *audience* fields, but would not match any records when the fields are removed.

4.4 Experimental results

Table 2 shows the performance of our model (SRM) against the three baselines. The model parameters were tuned using the 64 training queries on the *training* and *held-out* sets. The results are for the 63 test queries run against the *evaluation* corpus. (Similar results occur if the 64 training queries are run against

[9]This definition of relevance is unduly conservative by the standards of Information Retrieval researchers. Many records that might be considered relevant by a human annotator will be treated as non-relevant, artificially decreasing the accuracy of any retrieval algorithm. However, our approach has the advantage of being fully automatic: it allows us to test our model on a scale that would be prohibitively expensive with manual relevance judgments.

	cLM	bMatch	bLM	SRM	%change	improved
Rel-ret:	949	582	914	861	-5.80	26/50
Interpolated Recall - Precision:						
at 0.00	0.3852	0.3730	0.4153	0.5448	31.2	**33/49**
at 0.10	0.3014	0.3020	0.3314	0.4783	44.3	**42/56**
at 0.20	0.2307	0.2256	0.2660	0.3641	36.9	**40/59**
at 0.30	0.2105	0.1471	0.2126	0.2971	39.8	**36/58**
at 0.40	0.1880	0.1130	0.1783	0.2352	31.9	**36/58**
at 0.50	0.1803	0.0679	0.1591	0.1911	20.1	32/57
at 0.60	0.1637	0.0371	0.1242	0.1439	15.8	27/51
at 0.70	0.1513	0.0161	0.1001	0.1089	8.7	21/42
at 0.80	0.1432	0.0095	0.0901	0.0747	-17.0	18/36
at 0.90	0.1292	0.0055	0.0675	0.0518	-23.2	12/27
at 1.00	0.1154	0.0043	0.0593	0.0420	-29.2	9/23
Avg.Prec.	0.1790	0.1050	0.1668	0.2156	29.25	**43/63**
Precision at:						
5 docs	0.1651	0.2159	0.2413	0.3556	47.4	**32/43**
10 docs	0.1571	0.1651	0.2063	0.2889	40.0	**34/48**
15 docs	0.1577	0.1471	0.1841	0.2360	28.2	**32/49**
20 docs	0.1540	0.1349	0.1722	0.2024	17.5	28/47
30 docs	0.1450	0.1101	0.1492	0.1677	12.4	29/50
100 docs	0.0913	0.0465	0.0849	0.0871	2.6	**37/57**
200 docs	0.0552	0.0279	0.0539	0.0506	-6.2	33/53
500 docs	0.0264	0.0163	0.0255	0.0243	-4.5	26/48
1000 docs	0.0151	0.0092	0.0145	0.0137	-5.8	26/50
R-Prec.	0.1587	0.1204	0.1681	0.2344	39.44	**31/49**

Table 2: Performance of the 63 test queries retrieving 1000 documents on the evaluation data. Bold figures show statistically significant differences. Across all 63 queries, there are 1253 relevant documents.

the *evalution* corpus.)

The upper half of Table 2 shows precision at fixed recall levels; the lower half shows precision at different ranks. The *%change* column shows relative difference between our model and the baseline bLM. The *improved* column shows the number of queries where SRM exceeded bLM vs. the number of queries where performance was different. For example, $33/49$ means that SRM out-performed bLM on 33 queries out of 63, underperformed on $49{-}33{=}16$ queries, and had exactly the same performance on $63{-}49{=}14$ queries. Bold figures indicate statistically significant differences (according to the sign test with $p < 0.05$).

The results show that SRM outperforms three baselines in the high-precision region, beating bLM's mean average precision by 29%. User-oriented metrics, such as R-precision and precision at 10 documents, are improved by 39.4% and 44.3% respectively. The absolute performance figures are also very encouraging. Precision of 28% at rank 10 means that on average almost 3 out of the top 10 records in the ranked list are relevant, despite the requested fields not being available to the model.

We note that SRM continues to outperform bLM until very high recall and until the 100-document cutoff. After that, SRM degrades rapidly with respect to bLM. We feel the drop in effectiveness is of marginal interest because precision is already well below 10% and few users will be continuing to that depth in the list.

It is encouraging to see that SRM outperforms both cLM, the cheating baseline that takes advantage of the field values that are supposed to be "missing", and bMatch, suggesting that best-match retrieval provides a superior strategy for selecting a set of appropriate training records.

5 Conclusions

We have developed and empirically validated a new retrieval model for semi-structured text. The model is based on the idea that missing or corrupted values for one field can be inferred from values in other fields of the record. The cross-field inference makes it possible to find documents in response to a structured query when those query fields do not exist in the relevant documents at all.

We validated the SRM approach on a large archive of the NSDL repository. We developed a large set of structured Boolean queries that had relevant documents in the test portion of collection. We then indexed the documents *without* the fields used in the queries. As a result, using standard field matching approaches, not a single document would be returned in response to the queries—in particular, no relevant documents would be found.

We showed that standard information retrieval techniques and structured field matching could be combined to address this problem, but that the SRM approach outperforms them. We note that SRM brought two relevant documents into the top five—again, querying on missing fields—and achieved an average precision of 23%, a more than 35% improvement over a state-of-the-art relevance model approach combining the standard field matching.

Our work is continuing by exploring methods for handling fields with incorrect or corrupted values. The challenge becomes more than just inferring what values might be there; it requires combining likely missing values with confidence in the values already present: if an audience field contains 'undergraduate', it should be unlikely that 'K-6' would be a plausible value, too.

In addition to using SRMs for retrieval, we are currently extending the ideas to provide field validation and suggestions for data entry and validation: the same ideas used to find documents with missing field values can also be used to suggest potential values for a field and to identify values that seem inappropriate. We have also begun explorations toward using inferred values to help a user browse when starting from some structured information—e.g., given values for two fields, what values are probable for other fields.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

D.C. Blair. 1988. An extended relational document retrieval model. *Inf. Process. Manage.*, 24(3):349–371.

W.W. Cohen. 2000. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118(1–2):163–196.

S. DeFazio, A. Daoud, L. A. Smith, and J. Srinivasan. 1995. Integrating IR and RDBMS Using Cooperative Indexing. In *Proceedings of SIGIR*, pages 84–92.

B. C. Desai, P. Goyal, and F. Sadri. 1987. Non-first normal form universal relations: an application to information retrieval systems. *Inf. Syst.*, 12(1):49–55.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. 1999. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309.

N. Fuhr. 1993. A probabilistic relational model for the integration of IR and databases. In *Proceedings of SIGIR*, pages 309–317.

D. Heckerman, C. Meek, and D. Koller. 2004. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research.

J. Lafferty and C. Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR*, pages 111–119.

I. Macleod. 1991. Text retrieval and the relational model. *Journal of the American Society for Information Science*, 42(3):155–165.

J. Neville, D. Jensen, L. Friedland, and M. Hay. 2003. Learning relational probability trees. In *Proceedings of ACM KDD*, pages 625–630, New York, NY, USA.

J. M. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281.

B. Taskar, E. Segal, and D. Koller. 2001. Probabilistic classification and clustering in relational data. In *Proceedings of IJCAI*, pages 870–876.

S. R. Vasanthakumar, J.P. Callan, and W.B. Croft. 1996. Integrating INQUERY with an RDBMS to support text retrieval. *IEEE Data Eng. Bull.*, 19(1):24–33.

A.D. Vries and A. Wilschut. 1999. On the integration of IR and databases. In *Proceedings of IFIP 2.6 Working Conf. on Data Semantics*, Rotorua, New Zealand.

Improving Diversity in Ranking using Absorbing Random Walks

Xiaojin Zhu **Andrew B. Goldberg** **Jurgen Van Gael** **David Andrzejewski**
Department of Computer Sciences
University of Wisconsin, Madison
Madison, WI 53705
{jerryzhu, goldberg, jvangael, andrzeje}@cs.wisc.edu

Abstract

We introduce a novel ranking algorithm called GRASSHOPPER, which ranks items with an emphasis on diversity. That is, the top items should be different from each other in order to have a broad coverage of the whole item set. Many natural language processing tasks can benefit from such diversity ranking. Our algorithm is based on random walks in an absorbing Markov chain. We turn ranked items into absorbing states, which effectively prevents redundant items from receiving a high rank. We demonstrate GRASSHOPPER's effectiveness on extractive text summarization: our algorithm ranks between the 1st and 2nd systems on DUC 2004 Task 2; and on a social network analysis task that identifies movie stars of the world.

1 Introduction

Many natural language processing tasks involve ranking a set of items. Sometimes we want the top items to be not only good individually but also *diverse* collectively. For example, extractive text summarization generates a summary by selecting a few good sentences from one or more articles on the same topic (Goldstein et al., 2000). This can be formulated as ranking all the sentences, and taking the top ones. A good sentence is one that is representative, i.e., similar to many other sentences, so that it likely conveys the central meaning of the articles. On the other hand, we do not want multiple near-identical sentences. The top sentences should be diverse.

As another example, in information retrieval on news events, an article is often published by multiple newspapers with only minor changes. It is undesirable to rank all copies of the same article highly, even though it may be the most relevant. Instead, the top results should be different and complementary. In other words, one wants 'subtopic diversity' in retrieval results (Zhai et al., 2003).

The need for diversity in ranking is not unique to natural language processing. In social network analysis, people are connected by their interactions, e.g., phone calls. Active groups of people have strong interactions among them, but many groups may exist with fewer interactions. If we want a list of people that represent various groups, it is important to consider both activity and diversity, and not to fill the list with people from the same active groups.

Given the importance of diversity in ranking, there has been significant research in this area. Perhaps the most well-known method is maximum marginal relevance (MMR) (Carbonell and Goldstein, 1998), as well as cross-sentence informational subsumption (Radev, 2000), mixture models (Zhang et al., 2002), subtopic diversity (Zhai et al., 2003), diversity penalty (Zhang et al., 2005), and others. The basic idea is to penalize redundancy by lowering an item's rank if it is similar to items already ranked. However, these methods often treat centrality ranking and diversity ranking separately, sometimes with heuristic procedures.

We propose GRASSHOPPER (**G**raph **R**andom-walk with **A**bsorbing **S**tate**S** that **HOP**s among **PE**aks for **R**anking), a novel ranking algorithm that encourages diversity. GRASSHOPPER is an alternative to MMR and variants, with a principled mathematical model and strong empirical performance. It ranks a set of items such that: 1. A highly ranked item is representative of a local group in the set, i.e., it is similar to many other items (**centrality**); 2. The top items cover as many distinct groups as possible (**diversity**); 3. It incorporates an arbitrary pre-specified ranking as prior knowledge (**prior**). Importantly GRASSHOPPER achieves these in a unified framework of *absorbing Markov chain random walks*. The key idea is the following: We define a random walk on a graph over the items. Items which have been ranked so far become absorbing states. These absorbing states 'drag down' the importance of similar unranked states, thus encouraging diversity. Our model naturally balances centrality, diversity, and prior. We discuss the algorithm in Section 2. We present GRASSHOPPER's empirical results on text summarization and social network analysis in Section 3.

2 The GRASSHOPPER Algorithm

2.1 The Input

GRASSHOPPER requires three inputs: a graph W, a probability distribution $\mathbf{r}$ that encodes the prior ranking, and a weight $\lambda \in [0, 1]$ that balances the two.

The user needs to supply a graph with n nodes, one for each item. The graph is represented by an $n \times n$ weight matrix W, where w_{ij} is the weight on the edge from i to j. It can be either directed or undirected. W is symmetric for undirected graphs. The weights are non-negative. The graph does not need to be fully connected: if there is no edge from item i to j, then $w_{ij} = 0$. Self-edges are allowed. For example, in text summarization one can create an undirected, fully connected graph on the sentences. The edge between sentences i, j has weight w_{ij}, their cosine similarity. In social network analysis one can create a directed graph with w_{ij} being the number of phone calls i made to j. The graph should be constructed carefully to reflect domain knowledge. For examples, see (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Pang and Lee, 2004).

The user can optionally supply an arbitrary ranking on the items as prior knowledge. In this case GRASSHOPPER can be viewed as a re-ranking method. For example, in information retrieval, the prior ranking can be the ranking by relevance scores. In text summarization, it can be the position of sentences in the original article. (There is evidence that the first few sentences in an article are likely good summaries.) Somewhat unconventionally, the prior ranking is represented as a probability distribution $\mathbf{r} = (r_1, \cdots, r_n)^\top$ such that $r_i \geq 0, \sum_{i=1}^{n} r_i = 1$. The highest-ranked item has the largest probability, the next item has smaller probability, and so on. A distribution gives the user more control. For example $\mathbf{r}_a = (0.1, 0.7, 0.2)^\top$ and $\mathbf{r}_b = (0.3, 0.37, 0.33)^\top$ both represent the same ranking of items 2, 3, 1, but with different strengths. When there is no prior ranking, one can let $\mathbf{r} = (1/n, \cdots, 1/n)^\top$, the uniform distribution.

2.2 Finding the First Item

We find the first item in GRASSHOPPER ranking by teleporting random walks. Imagine a random walker on the graph. At each step, the walker may do one of two things: with probability λ, she moves to a neighbor state[1] according to the edge weights; otherwise she is teleported to a random state according to the distribution $\mathbf{r}$. Under mild conditions (which are satisfied in our setting, see below), the stationary distribution of the random walk defines the visiting probabilities of the nodes. The states with large probabilities can be regarded as central items, an idea used in Google PageRank (Page et al., 1998) and other information retrieval systems (Kurland and Lee, 2005; Zhang et al., 2005), text summarization (Erkan and Radev, 2004), keyword extraction (Mihalcea and Tarau, 2004) and so on. Depending on λ, items high on the user-supplied prior ranking $\mathbf{r}$ may also have large stationary probabilities, which is a way to incorporate the prior ranking.

As an example, we created a toy data set with 300 points in Figure 1(a). There are roughly three groups with different densities. We created a fully connected graph on the data, with larger edge weights if points are closer[2]. Figure 1(b) shows the stationary distribution of the random walk on the graph.

[1]We use *state*, *node* and *item* interchangeably.

[2]We use $w_{ij} = \exp(-\|x_i - x_j\|^2/0.16)$, $\lambda = 1$.

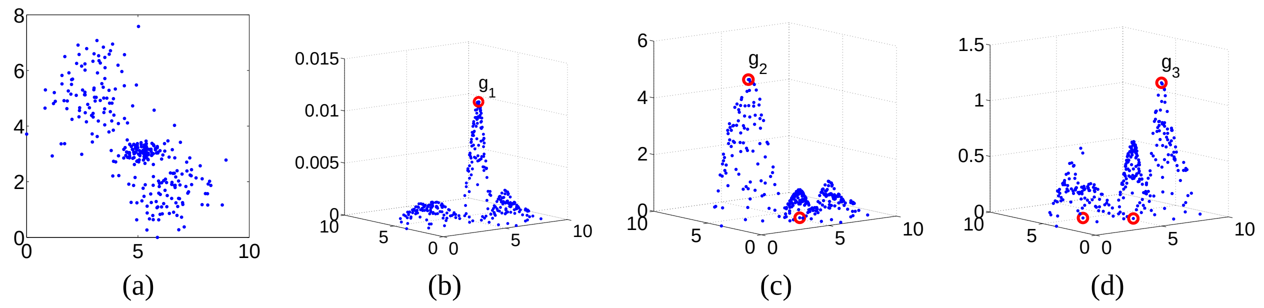


Figure 1: (a) A toy data set. (b) The stationary distribution π reflects centrality. The item with the largest probability is selected as the first item g_1. (c) The expected number of visits $\mathbf{v}$ to each node after g_1 becomes an absorbing state. (d) After both g_1 and g_2 become absorbing states. Note the diversity in g_1, g_2, g_3 as they come from different groups.

Items at group centers have higher probabilities, and tighter groups have overall higher probabilities.

However, the stationary distribution does not address diversity at all. If we were to rank the items by their stationary distribution, the top list would be dominated by items from the center group in Figure 1(b). Therefore we only use the stationary distribution to find the first item, and use a method described in the next section to rank the remaining items.

Formally we first define an $n \times n$ raw transition matrix $\tilde{P}$ by normalizing the rows of W: $\tilde{P}_{ij} = w_{ij}/\sum_{k=1}^{n} w_{ik}$, so that $\tilde{P}_{ij}$ is the probability that the walker moves to j from i. We then make the walk a teleporting random walk P by interpolating each row with the user-supplied initial distribution $\mathbf{r}$:

$$P = \lambda \tilde{P} + (1 - \lambda)\mathbf{1}\mathbf{r}^\top, \qquad (1)$$

where $\mathbf{1}$ is an all-1 vector, and $\mathbf{1}\mathbf{r}^\top$ is the outer product. If $\lambda < 1$ and $\mathbf{r}$ does not have zero elements, our teleporting random walk P is irreducible (possible to go to any state from any state by teleporting), aperiodic (the walk can return to a state after any number of steps), all states are positive recurrent (the expected return time to any state is finite) and thus ergodic (Grimmett and Stirzaker, 2001). Therefore P has a unique stationary distribution $\pi = P^\top \pi$. We take the state with the largest stationary probability to be the first item g_1 in GRASSHOPPER ranking: $g_1 = \operatorname{argmax}_{i=1}^{n} \pi_i$.

2.3 Ranking the Remaining Items

As mentioned early, the key idea of GRASSHOPPER is to turn ranked items into absorbing states. We first turn g_1 into an absorbing state. Once the random walk reaches an absorbing state, the walk is absorbed and stays there. It is no longer informative to compute the stationary distribution of an absorbing Markov chain, because the walk will eventually be absorbed. Nonetheless, it is useful to compute the *expected number of visits* to each node before absorption. Intuitively, those nodes strongly connected to g_1 will have many fewer visits by the random walk, because the walk tends to be absorbed soon after visiting them. In contrast, groups of nodes far away from g_1 still allow the random walk to linger among them, and thus have more visits. In Figure 1(c), once g_1 becomes an absorbing node (represented by a circle ‘on the floor’), the center group is no longer the most prominent: nodes in this group have fewer visits than the left group. Note now the y-axis is the number of visits instead of probability.

GRASSHOPPER selects the second item g_2 with the largest expected number of visits in this absorbing Markov chain. This naturally inhibits items similar to g_1 and encourages diversity. In Figure 1(c), the item near the center of the left group is selected as g_2. Once g_2 is selected, it is converted into an absorbing state, too. This is shown in Figure 1(d). The right group now becomes the most prominent, since both the left and center groups contain an absorbing state. The next item g_3 in ranking will come from the right group. Also note the range of y-axis is smaller:

with more absorbing states, the random walk will be absorbed sooner. The procedure is repeated until all items are ranked. The name GRASSHOPPER reflects the 'hopping' behavior on the peaks.

It is therefore important to compute the expected number of visits in an absorbing Markov chain. Let G be the set of items ranked so far. We turn the states $g \in G$ into absorbing states by setting $P_{gg} = 1$ and $P_{gi} = 0, \forall i \neq g$. If we arrange items so that ranked ones are listed before unranked ones, we can write P as

$$P = \left[\begin{array}{cc} \mathbf{I}_G & \mathbf{0} \\ R & Q \end{array} \right]. \tag{2}$$

Here $\mathbf{I}_G$ is the identity matrix on G. Submatrices R and Q correspond to rows of unranked items, those from (1). It is known that the *fundamental matrix*

$$N = (\mathbf{I} - Q)^{-1} \tag{3}$$

gives the expected number of visits in the absorbing random walk (Doyle and Snell, 1984). In particular N_{ij} is the expected number of visits to state j before absorption, if the random walk started at state i. We then average over all starting states to obtain v_j, the expected number of visits to state j. In matrix notation,

$$\mathbf{v} = \frac{N^\top \mathbf{1}}{n - |G|}, \tag{4}$$

where $|G|$ is the size of G. We select the state with the largest expected number of visits as the next item $g_{|G|+1}$ in GRASSHOPPER ranking:

$$g_{|G|+1} = \mathrm{argmax}_{i=|G|+1}^{n} \; v_i. \tag{5}$$

The complete GRASSHOPPER algorithm is summarized in Figure 2.

```
Input: $W$, $\mathbf{r}$, $\lambda$

  1. Create the initial Markov chain $P$ from
     $W, \mathbf{r}, \lambda$ (1).

  2. Compute $P$'s stationary distribution $\pi$. Pick the
     first item $g_1 = \mathrm{argmax}_i \; \pi_i$.

  3. Repeat until all items are ranked:

     (a) Turn ranked items into absorbing
         states (2).
     (b) Compute the expected number of visits $\mathbf{v}$
         for all remaining items (4). Pick the next
         item $g_{|G|+1} = \mathrm{argmax}_i \; v_i$
```

Figure 2: The GRASSHOPPER algorithm

2.4 Some Discussions

To see how λ controls the tradeoff, note when $\lambda = 1$ we ignore the user-supplied prior ranking $\mathbf{r}$, while when $\lambda = 0$ one can show that GRASSHOPPER returns the ranking specified by $\mathbf{r}$.

Our data in Figure 1(a) has a cluster structure. Many methods have exploited such structure, e.g., (Hearst and Pedersen, 1996; Leuski, 2001; Liu and Croft, 2004). In fact, a heuristic algorithm is to first cluster the items, then pick the central items from each cluster in turn. But it can be difficult to determine the appropriate number and control the shape of clusters. In contrast, GRASSHOPPER does *not* involve clustering. However it is still able to automatically take advantage of cluster structures in the data.

In each iteration we need to compute the fundamental matrix (3). This involves inverting an $(n - |G|) \times (n - |G|)$ matrix, which is expensive. However the Q matrix is reduced by one row and one column in every iteration, but is otherwise unchanged. This allows us to apply the matrix inversion lemma (Sherman-Morrison-Woodbury formula) (Press et al., 1992). Then we only need to invert the matrix once in the first iteration, but not in subsequent iterations. Space precludes a full discussion, but we point out that it presents a significant speed up. A Matlab implementation can be found at `http://www.cs.wisc.edu/~jerryzhu/pub/grasshopper.m`.

3 Experiments

3.1 Text Summarization

Multi-document extractive text summarization is a prime application for GRASSHOPPER. In this task, we must select and rank sentences originating from a set of documents about a particular topic or event. The goal is to produce a summary that includes all the relevant facts, yet avoids repetition that may result from using similar sentences from multiple documents. In this section, we demonstrate that

GRASSHOPPER's balance of centrality and diversity makes it successful at this task. We present empirical evidence that GRASSHOPPER achieves results competitive with the top text summarizers in the 2004 Document Understanding Conference (`http://duc.nist.gov`). DUC is a yearly text summarization community evaluation, with several tasks in recent years concentrating on multi-document summarization (described in more detail below).

Many successful text summarization systems achieve a balance between sentence centrality and diversity in a two-step process. Here we review the LexRank system (Erkan and Radev, 2004), which is most similar to our current approach. LexRank works by placing sentences in a graph, with edges based on the lexical similarity between the sentences (as determined by a cosine measure). Each sentence is then assigned a centrality score by finding its probability under the stationary distribution of a random walk on this graph. Unlike the similar PageRank algorithm (Page et al., 1998), LexRank uses an undirected graph of sentences rather than Web pages, and the edge weights are either cosine values or 0/1 with thresholding. The LexRank centrality can be combined with other centrality measures, as well as sentence position information. After this first step of computing centrality, a second step performs re-ranking to avoid redundancy in the highly ranked sentences. LexRank uses cross-sentence informational subsumption (Radev, 2000) to this end, but MMR (Carbonell and Goldstein, 1998) has also been widely used in the text summarization community. These methods essentially disqualify sentences that are too lexically similar to sentences ranked higher by centrality. In short, similar graph-based approaches to text summarization rely on two distinct processes to measure each sentence's importance and ensure some degree of diversity. GRASSHOPPER, on the other hand, achieves the same goal in a unified procedure.

We apply GRASSHOPPER to text summarization in the following manner. Our graph contains nodes for all the sentences in a document set. We used the Clair Library (`http://tangra.si.umich.edu/clair/clairlib`) to split documents into sentences, apply stemming, and create a cosine matrix for the stemmed sentences. Cosine values are computed using TF-IDF vectors. As in LexRank, edges in the graph correspond to text similarity. To create a sparse graph, we use the cosine threshold value of 0.1 obtained in (Erkan and Radev, 2004). Specifically, the edge weight between sentence vectors s_i and s_j is defined as

$$w_{ij} = \begin{cases} 1 & \text{if } \frac{s_i^\top s_j}{\|s_i\|\cdot\|s_j\|} > 0.1 \\ 0 & \text{otherwise} \end{cases} . \qquad (6)$$

The second input for GRASSHOPPER is an initial ranking distribution, which we derive from the position of each sentence in its originating document. Position forms the basis for lead-based summaries (i.e., using the first N sentences as the summary) and leads to very competitive summaries (Brandow et al., 1995). We form an initial ranking for each sentence by computing $p^{-\alpha}$, where p is the position of the sentence in its document, and α is a positive parameter trained on a development dataset. We then normalize over all sentences in all documents to form a valid distribution $r \propto p^{-\alpha}$ that gives high probability to sentences closer to the beginning of documents. With a larger α, the probability assigned to later sentences decays more rapidly.

To evaluate GRASSHOPPER, we experimented with DUC datasets. We train our parameters (α and λ) using the DUC 2003 Task 2 data. This dataset contains 30 document sets, each with an average of 10 documents about a news event. We test GRASSHOPPER's performance on the DUC 2004 Task 2, Tasks 4a and 4b data. DUC 2004 Task 2 has 50 document sets of 10 documents each. Tasks 4a and 4b explored cross-lingual summarization. These datasets consist of Arabic-to-English translations of news stories. The documents in Task 4a are machine-translated, while Task 4b's are manually-translated. Note that we handle the translated documents in exactly the same manner as the English documents.

We evaluate our results using the standard text summarization metric ROUGE (`http://www.isi.edu/~cyl/ROUGE/`). This is a recall-based measure of text co-occurrence between a machine-generated summary and model summaries manually created by judges. ROUGE metrics exist based on bigram, trigram, and 4-gram overlap, but ROUGE-1 (based on unigram matching) has been found to correlate best with human judgments (Lin and Hovy, 2003).

Using the DUC 2003 training data, we tuned α and λ on a small grid ($\alpha \in \{0.125, 0.25, 0.5, 1.0\}$; $\lambda \in \{0.0, 0.0625, 0.125, 0.25, 0.5, 0.95\}$). Specifically, for each of the 30 DUC 2003 Task 2 document sets, we computed ROUGE-1 scores comparing our generated summary to 4 model summaries. We averaged the resulting ROUGE-1 scores across all 30 sets to produce a single average ROUGE-1 score to assess a particular parameter configuration. After examining the results for all 24 configurations, we selected the best one: $\alpha = 0.25$ and $\lambda = 0.5$.

Table 1 presents our results using these parameter values to generate summaries for the three DUC 2004 datasets. Note that the averages listed are actually averages over 4 model summaries per set, and over all the sets. Following the standard DUC protocol, we list the confidence intervals calculated by ROUGE using a bootstrapping technique. The final column compares our results to the official systems that participated in the DUC 2004 evaluation. GRASSHOPPER is highly competitive in these text summarization tasks: in particular it ranks between the 1st and 2nd automatic systems on 2004 Task 2. The lower performance in Task 4a is potentially due to the documents being machine-translated. If they contain poorly translated sentences, graph edges based on cosine similarity could be less meaningful. For such a task, more advanced text processing is probably required.

3.2 Social Network Analysis

As another application of GRASSHOPPER, we identify the nodes in a social network that are the most prominent, and at the same time maximally cover the network. A node's prominence comes from its intrinsic stature, as well as the prominence of the nodes it touches. However, to ensure that the top-ranked nodes are representative of the larger graph structure, it is important to make sure the results are not dominated by a small group of highly prominent nodes who are closely linked to one another. This requirement makes GRASSHOPPER a useful algorithm for this task.

We created a dataset from the Internet Movie Database (IMDb) that consists of all comedy movies produced between 2000 and 2006, and have received more than 500 votes by IMDb users. This results in 1027 movies. We form a social network of actors by co-star relationship. Not surprisingly, actors from the United States dominate our dataset, although a total of 30 distinct countries are represented. We seek an actor ranking such that the top actors are prominent. However, we also want the top actors to be diverse, so they represent comedians from around the world.

This problem is framed as a GRASSHOPPER ranking problem. For each movie, we considered only the main stars, i.e., the first five cast members, who tend to be the most important. The resulting list contains 3452 unique actors. We formed a social network where the nodes are the actors, and undirected weighted edges connect actors who have appeared in a movie together. The edge weights are equal to the number of movies from our dataset in which both actors were main stars. Actors are also given a self-edge with weight 1. The co-star graph is given to GRASSHOPPER as an input. For the prior actor ranking, we simply let r be proportional to the number of movies in our dataset in which an actor has appeared. We set the weight $\lambda = 0.95$. It is important to note that no country information is ever given to GRASSHOPPER.

We use two measurements, 'country coverage' and 'movie coverage', to study the diversity and prominence of the ranking produced by GRASSHOPPER. We compare GRASSHOPPER to two baselines: ranking based solely on the number of movies an actor has appeared in, MOVIECOUNT, and a randomly generated ranking, RANDOM.

First, we calculate 'country coverage' as the number of different countries represented by the top k actors, for all k values. Each actor represents a single country—the country that the actor has appeared in the most. We hypothesize that actors are more likely to have co-star connections to actors within the same country, so our social network may have, to some extent, a clustering structure by country. 'Country coverage' approximates the number of clusters represented at different ranks.

Figure 3(a) shows that country coverage grows much more rapidly for GRASSHOPPER than for MOVIECOUNT. That is, we see more comedians from around the world ranked highly by GRASSHOPPER. In contrast, the top ranks of MOVIECOUNT are dominated by US actors, due to the relative abundance of US movies on IMDb. Many other countries are

Dataset	Number of Doc. Sets	Average ROUGE-1	95% C.I.	GRASSHOPPER Unofficial Rank
DUC 2004 Task 2	50	0.3755	[0.3622, 0.3888]	Between 1 & 2 of 34
DUC 2004 Task 4a	24	0.3785	[0.3613, 0.3958]	Between 5 & 6 of 11
DUC 2004 Task 4b	24	0.4067	[0.3883, 0.4251]	Between 2 & 3 of 11

Table 1: Text summarization results on DUC 2004 datasets. GRASSHOPPER was configured using parameters tuned on the DUC 2003 Task 2 dataset. The rightmost column lists what our rank would have been if we had participated in the DUC 2004 evaluation.

not represented until further down in the ranked list. This demonstrates that GRASSHOPPER ranking is successful in returning a more diverse ranking. Because of the absorbing states in GRASSHOPPER, the first few highly ranked US actors encourage the selection of actors from other regions of the co-star graph, which roughly correspond to different countries. RANDOM achieves even higher country coverage initially, but is quickly surpassed by GRASSHOPPER. The initial high coverage comes from the random selection of actors. However these randomly selected actors are often not prominent, as we show next.

Second, we calculate 'movie coverage' as the total number of unique movies the top k actors are in. We expect that actors who have been in more movies are more prominent. This is reasonable because we count an actor in a movie only if the actor is among the top five actors from that movie. Our counts thus exclude actors who had only small roles in numerous movies. Therefore high movie coverage roughly corresponds to ranking more prominent actors highly. It is worth noting that this measure also partially accounts for diversity, since an actor whose movies completely overlap with those of higher-ranked actors contributes nothing to movie coverage (i.e., his/her movies are already covered by higher-ranked actors).

Figure 3(b) shows that the movie coverage of GRASSHOPPER grows more rapidly than MOVIECOUNT, and much more rapidly than RANDOM. The results show that, while the RANDOM ranking is diverse, it is not of high quality because it fails to include many prominent actors in its high ranks. This is to be expected of a random ranking. Since the vast majority of the actors appear in only one movie, the movie coverage curve is roughly linear in the number of actors. By ranking more prominent actors highly, the GRASSHOPPER and MOVIECOUNT movie coverage curves grow faster. Many of the US actors highly ranked by MOVIECOUNT are co-stars of one another, so GRASSHOPPER outperforms MOVIECOUNT in terms of movie coverage too.

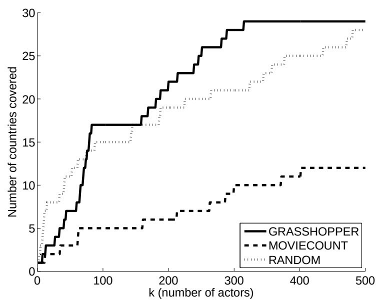
We inspect the GRASSHOPPER ranking, and find the top 5 actors to be Ben Stiller, Anthony Anderson, Johnny Knoxville, Eddie Murphy and Adam Sandler. GRASSHOPPER also brings many countries, and major stars from those countries, into the high ranks. Examples include Mads Mikkelsen ("synonym to the great success the Danish film industry has had"), Cem Yilmaz ("famous Turkish comedy actor, caricaturist and scenarist"), Jun Ji-Hyun ("face of South Korean cinema"), Tadanobu Asano ("Japan's answer to Johnny Depp"), Aamir Khan ("prominent Bollywood film actor"), and so on[3]. These actors are ranked significantly lower by MOVIECOUNT.

These results indicate that GRASSHOPPER achieves both prominence and diversity in ranking actors in the IMDb co-star graph.

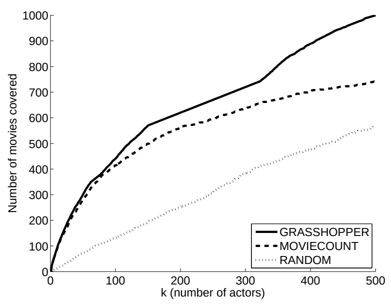
4 Conclusions

GRASSHOPPER ranking provides a unified approach for achieving both diversity and centrality. We have shown its effectiveness in text summarization and social network analysis. As future work, one direction is "partial absorption," where at each absorbing state the random walk has an escape probability to continue the random walk instead of being absorbed. Tuning the escape probability creates a continuum between PageRank (if the walk always escapes) and GRASSHOPPER (if always absorbed). In addition, we will explore the issue of parameter learning, and

[3]Quotes from IMDb and Wikipedia.



(a) Country coverage



(b) Movie coverage

Figure 3: (a) Country coverage at ranks up to 500, showing that GRASSHOPPER and RANDOM rankings are more diverse than MOVIECOUNT. (b) Movie coverage at ranks up to 500, showing that GRASSHOPPER and MOVIECOUNT have more prominent actors than RANDOM. Overall, GRASSHOPPER is the best.

user feedback (e.g., "This item should be ranked higher."). We also plan to apply GRASSHOPPER to a variety of tasks, including information retrieval (for example ranking news articles on the same event as in Google News, where many newspapers might use the same report and thus result in a lack of diversity), image collection summarization, and social network analysis for national security and business intelligence.

Acknowledgment We thank Mark Craven and the anonymous reviewers for helpful comments. This work is supported in part by Wisconsin Alumni Research Foundation (WARF) and NLM training grant 5T15LM07359.

References

R. Brandow, K. Mitze, and Lisa F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Inf. Process. Manage.*, 31(5):675–685.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*.

P.G. Doyle and J.L. Snell. 1984. *Random Walks and Electric Networks*. Mathematical Assoc. of America.

Güneş Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic summarization*, pages 40–48.

Geoffrey R. Grimmett and David R. Stirzaker. 2001. *Probability and Random Processes*. Oxford Science Publications, third edition.

Marti A. Hearst and Jan O. Pedersen. 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR-96*.

Oren Kurland and Lillian Lee. 2005. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *SIGIR'05*.

Anton Leuski. 2001. Evaluating document clustering for interactive information retrieval. In *CIKM'01*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL'03*, pages 71–78.

Xiaoyong Liu and W. Bruce Croft. 2004. Cluster-based retrieval using language models. In *SIGIR'04*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *EMNLP'04*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, pages 271–278.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. 1992. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press New York, NY, USA.

Dragomir Radev. 2000. A common theory of information fusion from multiple text sources, step one: Cross-document structure. In *Proceedings of the 1st ACL SIGDIAL Workshop on Discourse and Dialogue*.

ChengXiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *SIGIR'03*.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *SIGIR'02*.

Benyu Zhang, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen, and Wei-Ying Ma. 2005. Improving web search results using affinity graph. In *SIGIR'05*.

A Random Text Model for the Generation of Statistical Language Invariants

Chris Biemann

NLP Dept., University of Leipzig
Johannisgasse 26
04103 Leipzig, Germany
biem@informatik.uni-leipzig.de

Abstract

A novel random text generation model is introduced. Unlike in previous random text models, that mainly aim at producing a Zipfian distribution of word frequencies, our model also takes the properties of neighboring co-occurrence into account and introduces the notion of sentences in random text. After pointing out the deficiencies of related models, we provide a generation process that takes neither the Zipfian distribution on word frequencies nor the small-world structure of the neighboring co-occurrence graph as a constraint. Nevertheless, these distributions emerge in the process. The distributions obtained with the random generation model are compared to a sample of natural language data, showing high agreement also on word length and sentence length. This work proposes a plausible model for the emergence of large-scale characteristics of language without assuming a grammar or semantics.

1 Introduction

G. K. Zipf (1949) discovered that if all words in a sample of natural language are arranged in decreasing order of frequency, then the relation between a word's frequency and its rank in the list follows a power-law. Since then, a significant amount of research in the area of quantitative linguistics has been devoted to the question how this property emerges and what kind of processes generate such Zipfian distributions.

The relation between the frequency of a word at rank r and its rank is given by $f(r) \propto r^{-z}$, where z is the exponent of the power-law that corresponds to the slope of the curve in a log plot (cf. figure 2). The exponent z was assumed to be exactly 1 by Zipf; in natural language data, also slightly differing exponents in the range of about 0.7 to 1.2 are observed (cf. Zanette and Montemurro 2002). B. Mandelbrot (1953) provided a formula with a closer approximation of the frequency distributions in language data, noticing that Zipf's law holds only for the medium range of ranks, whereas the curve is flatter for very frequent words and steeper for high ranks. He also provided a word generation model that produces random words of arbitrary average length in the following way: With a probability w, a word separator is generated at each step, with probability (1-w)/N, a letter from an alphabet of size N is generated, each letter having the same probability. This is sometimes called the "monkey at the typewriter" (Miller, 1957). The frequency distribution follows a power-law for long streams of words, yet the equiprobability of letters causes the plot to show a step-wise rather than a smooth behavior, as examined by Ferrer i Cancho and Solé (2002), cf. figure 2. In the same study, a smooth rank distribution could be obtained by setting the letter probabilities according to letter frequencies in a natural language text. But the question of how these letter probabilities emerge remains unanswered.

Another random text model was given by Simon (1955), which does not take an alphabet of single letters into consideration. Instead, at each time step, a previously unseen new word is added to the stream with a probability a, whereas with probability (1-a), the next word is chosen amongst the words at previous positions. As words with higher frequency in the already generated stream have a higher probability of being added again, this imposes a strong competition among different words, resulting in a frequency distribution that follows a power-law with exponent z=(1-a). This was taken up by Zanette and Montemurro (2002), who slightly modify Simon's model. They introduce sublinear vocabulary growth by additionally making the new word probability dependent on the time step. Furthermore, they introduce a threshold on the maximal probability a previously seen word can be assigned to for generation, being able to modify the exponent z as well as to model the flatter curve for high frequency words. In (Ha et al., 2002), Zipf's law is extended to words and phrases, showing its validity for syllable-class based languages when conducting the extension.

Neither the Mandelbrot nor the Simon generation model take the sequence of words into account. Simon treats the previously generated stream as a bag of words, and Mandelbrot does not consider the previous stream at all. This is certainly an over-simplification, as natural language exhibits structural properties within sentences and texts that are not grasped by bags of words.

The work by Kanter and Kessler (1995) is, to our knowledge, the only study to date that takes the word order into account when generating random text. They show that a 2-parameter Markov process gives rise to a stationary distribution that exhibits the word frequency distribution and the letter frequency distribution characteristics of natural language. However, the Markov process is initialized such that any state has exactly two successor states, which means that after each word, only two other following words are possible. This certainly does not reflect natural language properties, where in fact successor frequencies of words follow a power-law and more successors can be observed for more frequent words. But even when allowing a more realistic number of successor states, the transition probabilities of a Markov model need to be initialized a priori in a sensible way. Further, the fixed number of states does not allow for infinite vocabulary.

In the next section we provide a model that does not suffer from all these limitations.

2 The random text generation model

When constructing a random text generation model, we proceed according to the following guidelines (cf. Kumar et al. 1999 for web graph generation):

- *simplicity*: a generation model should reach its goal using the simplest mechanisms possible but results should still comply to characteristics of real language
- *plausibility*: Without claiming that our model is an exhaustive description of what makes human brains generate and evolve language, there should be at least a possibility that similar mechanisms could operate in human brains. For a discussion on the sensitivity of people to bigram statistics, see e.g. (Thompson and Newport, 2007).
- *emergence*: Rather than constraining the model with the characteristics we would like to see in the generated stream, these features should emerge in the process.

Our model is basically composed of two parts that will be described separately: A word generator that produces random words composed of letters and a sentence generator that composes random sentences of words. Both parts use an internal graph structure, where traces of previously generated words and sentences are memorized. The model is inspired by small-world network generation processes, cf. (Watts and Strogatz 1998, Barabási and Albert 1999, Kumar et al. 1999, Steyvers and Tenenbaum 2005). A key notion is the strategy of following beaten tracks: Letters, words and sequences of words that have been generated before are more likely to be generated again in the future - a strategy that is only fulfilled for words in Simon's model.

But before laying out the generators in detail, we introduce ways of testing agreement of our random text model with natural language text.

2.1 Testing properties of word streams

All previous approaches aimed at reproducing a Zipfian distribution on word frequency, which is a criterion that we certainly have to fulfill. But there are more characteristics that should be obeyed to make a random text more similar to natural language than previous models:

- *Lexical spectrum*: The smoothness or stepwise shape of the rank-frequency distribution affects the lexical spectrum, which is the probability distribution on word frequency. In natural language texts, this distribution follows a power-law with an exponent close to 2 (cf. Ferrer i Cancho and Solé, 2002).

- *Distribution of word length:* According to (Sigurd et al., 2004), the distribution of word frequencies by length follows a variant of the gamma distribution
- *Distribution of sentence length*: The random text's sentence length distribution should resemble natural language. In (Sigurd et al., 2004), the same variant of the gamma distribution as for word length is fit to sentence length.
- *Significant neighbor-based co-occurrence*: As discussed in (Dunning 1993), it is possible to measure the amount of surprise to see two neighboring words in a corpus at a certain frequency under the assumption of independence. At random generation without word order awareness, the number of such pairs that are significantly co-occurring in neighboring positions should be very low. We aim at reproducing the number of significant pairs in natural language as well as the graph structure of the neighbor-based co-occurrence graph.

The last characteristic refers to the distribution of words in sequence. Important is the notion of significance, which serves as a means to distinguish random sequences from motivated ones. We use the log-likelihood ratio for determining significance as in (Dunning, 1993), but other measures are possible as well. Note that the model of Kanter and Kessler (1995) produces a maximal degree of 2 in the neighbor-based co-occurrence graph.

As written language is rather an artifact of the most recent millennia then a realistic sample of everyday language, we use the beginning of the spoken language section of the British National Corpus (BNC) to test our model against. For simplicity, all letters are capitalized and special characters are removed, such that merely the 26 letters of the English alphabet are contained in the sample. Being aware that a letter transcription is in itself an artifact of written language, we chose this as a good-enough approximation, although operating on phonemes instead of letters would be preferable. The sample contains 1 million words in 125,395 sentences with an average length of 7.975 words, which are composed of 3.502 letters in average.

2.2 Basic notions of graph theory

As we use graphs for the representation of memory in both parts of the model, some basic notions of graph theory are introduced. A graph G(V,E) consists of a set of vertices V and a set of weighted, directed edges between two vertices $E \subset V \times V \times R$ with R real numbers. The first vertex of an edge is called startpoint, the second vertex is called endpoint. A function weight: $V \times V \rightarrow R$ returns the weight of edges. The indegree (outdegree) of a vertex v is defined as the number of edges with v as startpoint (endpoint). The degree of a vertex is equal to its indegree and outdegree if the graph is undirected, i.e. $(u,v,w) \in E$ implies $(v,u,w) \in E$. The neighborhood neigh(v) of a vertex v is defined as the set of vertices $s \in S$ where $(v,s,weight(v,s)) \in E$.

The clustering coefficient is the probability that two neighbors X and Y of a given vertex Z are themselves neighbors, which is measured for undirected graphs (Watts and Strogatz, 1998). The amount of existing edges amongst the vertices in the neighborhood of a vertex v is divided by the number of possible edges. The average over all vertices is defined as the clustering coefficient C.

The small-world property holds if the average shortest path length between pairs of vertices is comparable to a random graph (Erdös and Rényi, 1959), but its clustering coefficient is much higher. A graph is called scale-free (cf. Barabási and Albert, 1999), if the degree distribution of vertices follows a power-law.

2.3 Word Generator

The word generator emits sequences of letters, which are generated randomly in the following way: The word generator starts with a graph of all N letters it is allowed to choose from. Initially, all vertices are connected to themselves with weight 1. When generating a word, the generator chooses a letter x according to its probability P(x), which is computed as the normalized weight sum of outgoing edges:

$$P(x) = \frac{weightsum(x)}{\sum_{v \in V} weightsum(v)}$$

$$weightsum(y) = \sum_{u \in neigh(y)} weight(y,u).$$

After the generation of the first letter, the word generator proceeds with the next position. At every position, the word ends with a probability w∈(0,1) or generates a next letter according to the letter production probability as given above. For every letter bigram, the weight of the directed edge between the preceding and current letter in the letter graph is increased by one. This results in self-reinforcement of letter probabilities: the more often a letter is generated, the higher its weight sum will be in subsequent steps, leading to an increased generation probability. Figure 1 shows how a word generator with three letters A,B,C changes its weights during the generation of the words AA, BCB and ABC.

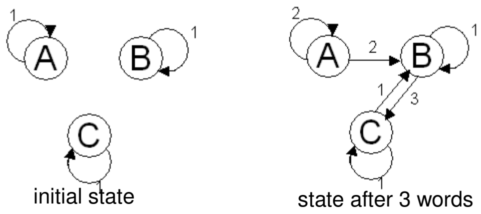


Figure 1: Letter graph of the word generator. Left: initial state. Right.: State after generating AA, BCB and ABC. The numbers next to edges are edge weights. The probability for the letters for the next step are P(A)=0.4, P(B)=0.4 and P(C)=0.2.

The word end probability w directly influences the average word length, which is given by 1+(1/w). For random number generation, we use the Mersenne Twister (Masumoto and Nishimura, 1998).

The word generator itself does produce a smooth Zipfian distribution on word frequencies and a lexical spectrum following a power-law. Figure 2 shows frequency distribution and lexical spectrum of 1 million words as generated by the word generator with w=0.2 on 26 letters in comparison to a Mandelbrot generator with the same parameters. The reader might note that a similar behaviour could be reached by just setting the probability of generating a letter according to its relative frequency in previously generated words. The graph seems an unnecessary complication for that reason. But retaining the letter graph with directed edges gives rise to model the sequence of letters for a more plausible morphological production in future extensions of this model, probably in a similar way than in the sentence generator as described in the following section.

As depicted in figure 2, the word generator fulfills the requirements on Zipf's law and the lexical spectrum, yielding a Zipfian exponent of around 1 and a power-law exponent of 2 for a large regime in the lexical spectrum, both matching the values as observed previously in natural language in e.g. (Zipf, 1949) and (Ferrer i Cancho and Solé, 2002). In contrast to this, the Mandelbrot model shows to have a step-wise rank-frequency distribution and a distorted lexical spectrum. Hence, the word generator itself is already an improvement over previous models as it produces a smooth Zipfian distribution and a lexical spectrum following a power-law. But to comply to the other requirements as given in section 2.1, the process has to be extended by a sentence generator.

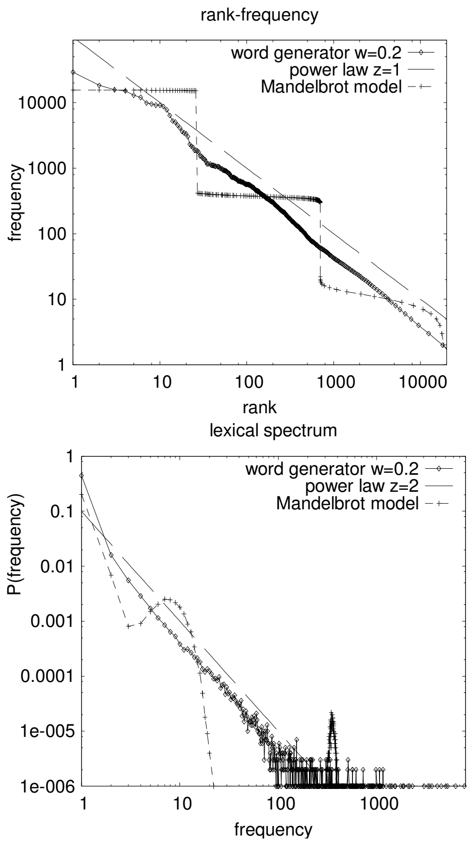


Figure 2: rank-frequency distribution and lexical spectrum for the word generator in comparison to the Mandelbrot model

2.4 Sentence Generator

The sentence generator model retains another directed graph, which memorizes words and their sequences. Here, vertices correspond to words and edge weights correspond to the number of times two words were generated in a sequence. The word graph is initialized with a begin-of-sentence (BOS) and an end-of-sentence (EOS) symbol, with an edge of weight 1 from BOS to EOS. When generating a sentence, a random walk on the directed edges starts at the BOS vertex. With a new word probability (1-s), an existing edge is followed from the current vertex to the next vertex according to its weight: the probability of choosing endpoint X from the endpoints of all outgoing edges from the current vertex C is given by

$$P(word = X) = \frac{weight(C,X)}{\sum_{N \in neigh(C)} weight(C,N)}.$$

Otherwise, with probability $s \in (0,1)$, a new word is generated by the word generator model, and a next word is chosen from the word graph in proportion to its weighted indegree: the probability of choosing an existing vertex E as successor of a newly generated word N is given by

$$P(word = E) = \frac{indgw(E)}{\sum_{v \in V} indgw(v)},$$

$$indgw(X) = \sum_{v \in V} weight(v,X).$$

For each sequence of two words generated, the weight of the directed edge between them is increased by 1. Figure 3 shows the word graph for generating in sequence: (empty sentence), AA, AA BC, AA, (empty sentence), AA CA BC AA, AA CA CA BC.

During the generation process, the word graph grows and contains the full vocabulary used so far for generating in every time step. It is guaranteed that a random walk starting from BOS will finally reach the EOS vertex. It can be expected that sentence length will slowly increase during the course of generation as the word graph grows and the random walk has more possibilities before finally arriving at the EOS vertex. The sentence length is influenced by both parameters of the model: the word end probability w in the word generator and the new word probability s in the sentence generator. By feeding the word transitions back into the generating model, a reinforcement of previously generated sequences is reached. Figure 4 illustrates the sentence length growth for various parameter settings of w and s.

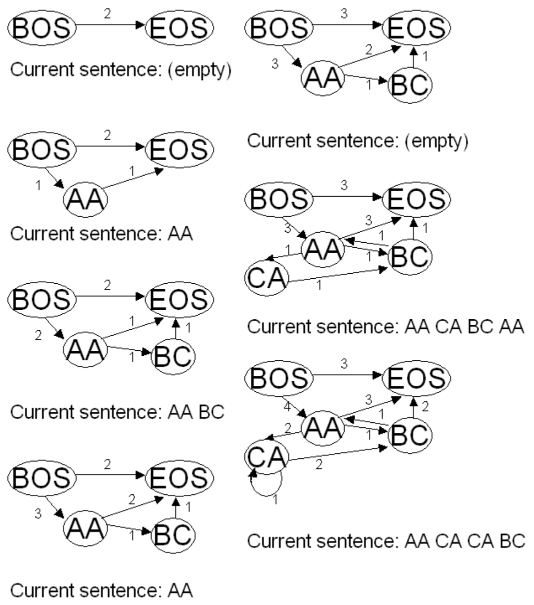

Current sentence: (empty)
Current sentence: AA
Current sentence: AA BC
Current sentence: AA
Current sentence: (empty)
Current sentence: AA CA BC AA
Current sentence: AA CA CA BC

Figure 3: the word graph of the sentence generator model. Note that in the last step, the second CA was generated as a new word from the word generator. The generation of empty sentences happens frequently. These are omitted in the output.

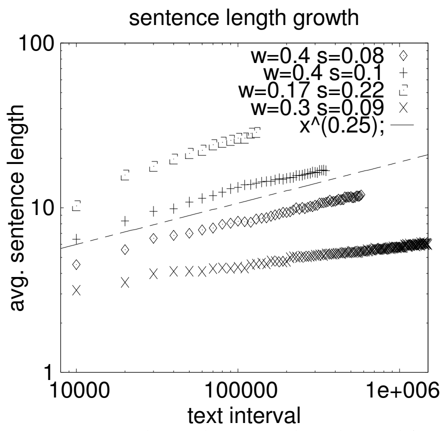


Figure 4: sentence length growth, plotted in average sentence length per intervals of 10,000 sentences. The straight line in the log plot indicates a polynomial growth.

It should be noted that the sentence generator produces a very diverse sequence of sentences which does not deteriorate in repeating the same sentence all over again in later stages. Both word and sentence generator can be viewed as weighted finite automata (cf. Allauzen et al., 2003) with self-training.

After having defined our random text generation model, the next section is devoted to testing it according to the criteria given in section 2.1.

3 Experimental results

To measure agreement with our BNC sample, we generated random text with the sentence generator using w=0.4 and N=26 to match the English average word length and setting s to 0.08 for reaching a comparable sentence length. The first 50,000 sentences were skipped to reach a relatively stable sentence length throughout the sample. To make the samples comparable, we used 1 million words totaling 125,345 sentences with an average sentence length of 7.977.

3.1 Word frequency

The comparison between English and the sentence generator w.r.t the rank-frequency distribution is depicted in figure 5.

Both curves follow a power-law with z close to 1.5, in both cases the curve is flatter for high frequency words as observed by Mandelbrot (1953). This effect could not be observed to this extent for the word generator alone (cf. figure 2).

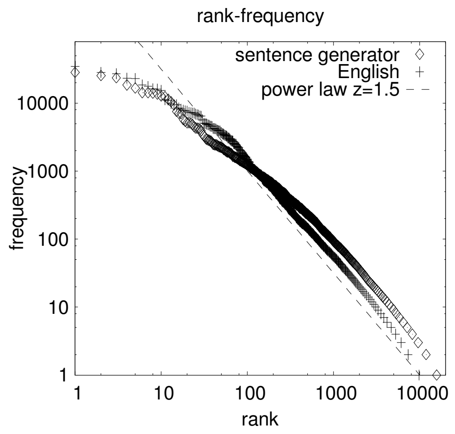


Figure 5: rank-frequency plot for English and the sentence generator

3.2 Word length

While the word length in letters is the same in both samples, the sentence generator produced more words of length 1, more words of length>10 and less words of medium length. The deviation in single letter words can be attributed to the writing system being a transcription of phonemes and few phonemes being expressed with only one letter. However, the slight quantitative differences do not oppose the similar distribution of word lengths in both samples, which is reflected in a curve of similar shape in figure 6 and fits well the gamma distribution variant of (Sigurd et al., 2004).

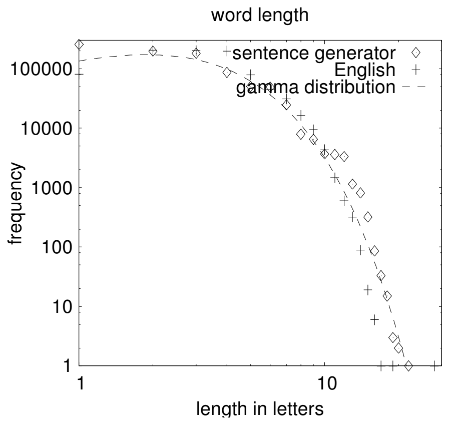


Figure 6: Comparison of word length distributions. The dotted line is the function as introduced in (Sigurd et al., 2004) and given by $f(x) \propto x^{1.5} \cdot 0.45^{x}$.

3.3 Sentence length

The comparison of sentence length distribution shows again a high capability of the sentence generator to model the distribution of the English sample. As can be seen in figure 7, the sentence generator produces less sentences of length>25 but does not show much differences otherwise. In the English sample, there are surprisingly many two-word sentences.

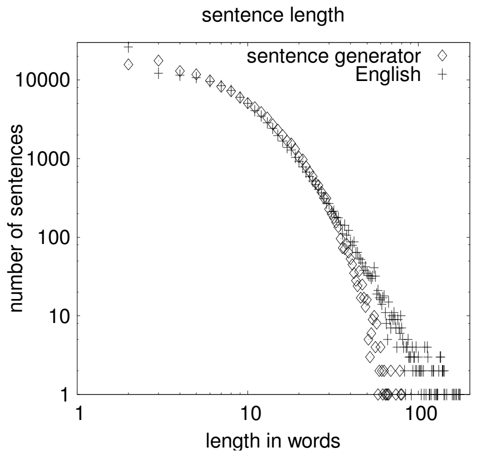


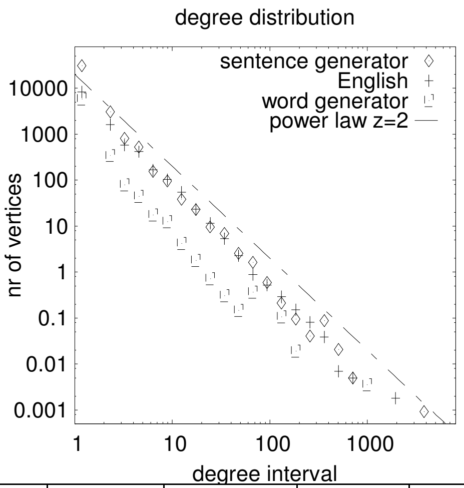
Figure 7: Comparison of sentence length distribution.

3.4 Neighbor-based co-occurrence

In this section, the structure of the significant neighbor-based co-occurrence graphs is examined.

The significant neighbor-based co-occurrence graph contains all words as vertices that have at least one co-occurrence to another word exceeding a certain significance threshold. The edges are undirected and weighted by significance. Ferrer i Cancho and Solé (2001) showed that the neighbor-based co-occurrence graph of the BNC is scale-free and the small-world property holds.

For comparing the sentence generator sample to the English sample, we compute log-likelihood statistics (Dunning, 1993) on neighboring words that at least co-occur twice. The significance threshold was set to 3.84, corresponding to 5% error probability when rejecting the hypothesis of mutual independence. For both graphs, we give the number of vertices, the average shortest path length, the average degree, the clustering coefficient and the degree distribution in figure 8. Further, the characteristics of a comparable random graph as defined by (Erdös and Rényi, 1959) are shown.



	English sample	sentence gen.	word gen.	random graph
# of ver.	7154	15258	3498	10000
avg. sht. path	2.933	3.147	3.601	4.964
avg. deg.	9.445	6.307	3.069	7
cl.coeff.	0.2724	0.1497	0.0719	6.89E-4
z	1.966	2.036	2.007	-

Figure 8: Characteristics of the neighbor-based co-occurrence graphs of English and the generated sample.

From the comparison with the random graph it is clear that both neighbor-based graphs exhibit the small-world property as their clustering coefficient is much higher than in the random graph while the average shortest path lengths are comparable. In quantity, the graph obtained from the generated sample has about twice as many vertices but its clustering coefficient is about half as high as in the English sample. This complies to the steeper rank-frequency distribution of the English sample (see fig. 5), which is, however, much steeper than the average exponent found in natural language. The degree distributions clearly match with a power-law exponent of 2, which does not confirm the two regimes of different slopes as in (Ferrer i Cancho and Solé 2001). The word generator data produced an number of significant co-occurrences that lies in the range of what can be expected from the 5% error of the statistical test. The degree distribution plot appears shifted downwards about one decade, clearly not matching the distribution of words in sequence of natural language.

Considering the analysis of the significant neighbor-based co-occurrence graph, the claim is supported that the sentence generator model reproduces the characteristics of word sequences in natural language on the basis of bigrams.

4 Conclusion

In this work we introduced a random text generation model that fits well with natural language with respect to frequency distribution, word length, sentence length and neighboring co-occurrence. The model was not constrained by any a priori distribution – the characteristics emerged from a 2-level process involving one parameter for the word generator and one parameter for the sentence generator. This is, to our knowledge, the first random text generator that models sentence boundaries beyond inserting a special blank character at random: rather, sentences are modeled as a path between sentence beginning and sentence end which imposes restrictions on the words possible at sentence beginnings and endings. Considering its simplicity, we have therefore proposed a plausible model for the emergence of large-scale characteristics of language without assuming a grammar or semantics. After all, our model produces gibberish – but gibberish that is well distributed.

The studies of Miller (1957) rendered Zipf's law un-interesting for linguistics, as it is a mere artifact of language rather than playing an important role in its production, as it emerges when putting a monkey in front of a typewriter. Our model does not only explain Zipf's law, but many other characteristics of language, which are obtained with a monkey that follows beaten tracks. These additional characteristics can be thought of as artifacts as well, but we strongly believe that the study of random text models can provide insights in the process that lead to the origin and the evolution of human languages.

For further work, an obvious step is to improve the word generator so that it produces morphologically more plausible sequences of letters and to intertwine both generators for the emergence of word categories. Furthermore, it is desirable to embed the random generator in models of communication where speakers parameterize language generation of hearers and to examine, which structures are evolutionary stable (see Jäger, 2003). This would shed light on the interactions between different levels of human communication.

Acknowledgements

The author would like to thank Colin Bannard, Reinhard Rapp and the anonymous reviewers for useful comments.

References

C. Allauzen, M. Mohri, and B. Roark. 2003. *Generalized algorithms for constructing language models*. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pp. 40–47

A.-L. Barabási and R. Albert. 1999. Emergence of scaling in random networks. *Science*, 286:509-512

T. Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1), pp. 61-74

P. Erdös and A. Rényi. 1959. *On Random Graphs I.* Publicationes Mathematicae (Debrecen)

R. Ferrer i Cancho and R. V. Solé. 2001. *The small-world of human language*. Proceedings of the Royal Society of London B 268 pp. 2261-2266

R. Ferrer i Cancho and R. V. Solé. 2002. Zipf's law and random texts. *Advances in Complex Systems*, Vol.5 No. 1 pp. 1-6

L. Q. Ha, E. Sicilia-Garcia, J. Ming and F.J. Smith. 2002. *Extension of Zipf's law to words and phrases*. Proceedings of 19th International Conference on Computational Linguistics (COLING-2002), pp. 315-320.

G. Jäger. 2003. *Evolutionary Game Theory and Linguistic Typology: A Case Study*. Proceedings of the 14th Amsterdam Colloquium, ILLC, University of Amsterdam, 2003.

I. Kanter and D. A. Kessler. 1995. Markov Processes: Linguistics and Zipf's law. *Physical review letters*, 74:22

S. R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. 1999. Extracting Large-Scale Knowledge Bases from the Web. *The VLDB Journal*, pp. 639-650

B. B. Mandelbrot. 1953. *An information theory of the statistical structure of language*. In Proceedings of the Symposium on Applications of Communications Theory, London

M. Matsumoto and T. Nishimura. 1998. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, Vol. 8, No. 1, pp.3-30

G. A. Miller. 1957. Some effects of intermittent silence,. *American Journal of Psychology*, 70, pp. 311-314

H. A. Simon. 1955. On a class of skew distribution functions. *Biometrika*, 42, pp. 425-440

B. Sigurd, M. Eeg-Olofsson and J. van de Weijer. 2004. word length, sentence length and frequency – Zipf revisited. *Studia Linguistica*, 58(1), pp. 37-52

M. Steyvers and J. B. Tenenbaum. 2005. The large-scale structure of semantic networks: statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1)

S. P. Thompson and E. L. Newport. 2007. Statistical learning of syntax: The role of transitional probability. *Language Learning and Development*, 3, pp. 1-42.

D. J. Watts and S. H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature*, 393 pp. 440-442

D. H. Zanette and M. A. Montemurro. 2002. *Dynamics of text generation with realistic Zipf distribution*. arXiv:cond-mat/0212496

G. K. Zipf. 1949. *Human Behavior and the Principle of least Effort*. Cambridge, MA: Addison Wesley

A Systematic Exploration of the Feature Space for Relation Extraction

Jing Jiang and **ChengXiang Zhai**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{jiang4,czhai}@cs.uiuc.edu

Abstract

Relation extraction is the task of finding semantic relations between entities from text. The state-of-the-art methods for relation extraction are mostly based on statistical learning, and thus all have to deal with feature selection, which can significantly affect the classification performance. In this paper, we systematically explore a large space of features for relation extraction and evaluate the effectiveness of different feature subspaces. We present a general definition of feature spaces based on a graphic representation of relation instances, and explore three different representations of relation instances and features of different complexities within this framework. Our experiments show that using only basic unit features is generally sufficient to achieve state-of-the-art performance, while over-inclusion of complex features may hurt the performance. A combination of features of different levels of complexity and from different sentence representations, coupled with task-oriented feature pruning, gives the best performance.

1 Introduction

An important information extraction task is relation extraction, whose goal is to detect and characterize semantic relations between entities in text. For example, the text fragment "hundreds of Palestinians converged on the square" contains the *located* relation between the *Person* entity "hundreds of Palestinians" and the *Bounded-Area* entity "the square". Relation extraction has applications in many domains, including finding affiliation relations from web pages and finding protein-protein interactions from biomedical literature.

Recent studies on relation extraction have shown the advantages of discriminative model-based statistical machine learning approach to this problem. There are generally two lines of work following this approach. The first utilizes a set of carefully selected features obtained from different levels of text analysis, from part-of-speech (POS) tagging to full parsing and dependency parsing (Kambhatla, 2004; Zhao and Grishman, 2005; Zhou et al., 2005)[1]. The second line of work designs kernel functions on some structured representation (sequences or trees) of the relation instances to capture the similarity between two relation instances (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005a; Bunescu and Mooney, 2005b; Zhang et al., 2006a; Zhang et al., 2006b). Of particular interest among the various kernels proposed are the convolution kernels (Bunescu and Mooney, 2005b; Zhang et al., 2006a), because they can efficiently compute the similarity between two instances in a huge feature space due to their recursive nature. Apart from their computational efficiency, convolution kernels also implicitly correspond to some feature space. Therefore, both lines of work rely on an appropriately de-

[1]Although Zhao and Grishman (2005) defined a number of kernels for relation extraction, the method is essentially similar to feature-based methods.

Proceedings of NAACL HLT 2007, pages 113–120,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

fined set of features. As in any learning problem, the choice of features can affect the performance significantly.

Despite the importance of feature selection, there has not been any systematic exploration of the feature space for relation extraction, and the choices of features in existing work are somewhat arbitrary. In this paper, we conduct a systematic study of the feature space for relation extraction, and evaluate the effectiveness of different feature subspaces. Our motivations are twofold. First, based on previous studies, we want to identify and characterize the types of features that are potentially useful for relation extraction, and define a relatively complete and structured feature space that can be systematically explored. Second, we want to compare the effectiveness of different features. Such a study can guide us to choose the most effective feature set for relation extraction, or to design convolution kernels in the most effective way.

We propose and define a unified graphic representation of the feature space, and experiment with three feature subspaces, corresponding to sequences, syntactic parse trees and dependency parse trees. Experiment results show that each subspace is effective by itself, with the syntactic parse tree subspace being the most effective. Combining the three subspaces does not generate much improvement. Within each feature subspace, using only the basic unit features can already give reasonably good performance. Adding more complex features may not improve the performance much or may even hurt the performance. Task-oriented heuristics can be used to prune the feature space, and when appropriately done, can improve the performance. A combination of features of different levels of complexity and from different sentence representations, coupled with task-oriented feature pruning, gives the best performance.

2 Related Work

Zhao and Grishman (2005) and Zhou et al. (2005) explored a large set of features that are potentially useful for relation extraction. However, the feature space was defined and explored in a somewhat ad hoc manner. We study a broader scope of features and perform a more systematic study of different feature subspaces. Zelenko et al. (2003) and Culotta and Sorensen (2004) used tree kernels for relation extraction. These kernels can achieve high precision but low recall because of the relatively strict matching criteria. Bunescu and Mooney (2005a) proposed a dependency path kernel for relation extraction. This kernel also suffers from low recall for the same reason. Bunescu and Mooney (2005b) and Zhang et. al. (2006a; 2006b) applied convolution string kernels and tree kernels, respectively, to relation extraction. The convolution tree kernels achieved state-of-the-art performance. Since convolution kernels correspond to some explicit large feature spaces, the feature selection problem still remains.

General structural representations of natural language data have been studied in (Suzuki et al., 2003; Cumby and Roth, 2003), but these models were not designed specifically for relation extraction. Our feature definition is similar to these models, but more specifically designed for relation extraction and systematic exploration of the feature space. Compared with (Cumby and Roth, 2003), our feature space is more compact and provides more guidance on selecting meaningful subspaces.

3 Task Definition

Given a small piece of text that contains two entity mentions, the task of relation extraction is to decide whether the text states some semantic relation between the two entities, and if so, classify the relation into one of a set of predefined semantic relation types. Formally, let $r = (s, arg_1, arg_2)$ denote a relation instance, where s is a sentence, arg_1 and arg_2 are two entity mentions contained in s, and arg_1 precedes arg_2 in the text. Given a set of relation instances $\{r_i\}$, each labeled with a type $t_i \in \mathcal{T}$, where $\mathcal{T}$ is the set of predefined relation types plus the type *nil*, our goal is to learn a function that maps a relation instance r to a type $t \in \mathcal{T}$. Note that we do not specify the representation of s here. Indeed, s can contain more structured information in addition to merely the sequence of tokens in the sentence.

4 Feature Space for Relation Extraction

Ideally, we would like to define a feature space with at least two properties: (1) It should be *complete* in the sense that all features potentially useful for the classification problem are included. (2) It should have a good *structure* so that a systematic search in the space is possible. Below we show how a unified graph-based feature space can be defined to satisfy these two properties.

4.1 A Unified View of Features for Relation Extraction

Before we introduce our definition of the feature space, let us first look at some typical features used for relation extraction. Consider the relation instance *"hundreds of Palestinians converged on the square"* with arg_1 = *"hundreds of Palestinians"* and arg_2 = *"the square"*. Various types of information can be useful for classifying this relation instance. For example, knowing that arg_1 is an entity of type *Person* can be useful. This feature involves the single token *"Palestinians"*. Another feature, "the head word of arg_1 (*Palestinians*) is followed by a verb (*converged*)", can also be useful. This feature involves two tokens, *"Palestinians"* and *"converged"*, with a sequence relation. It also involves the knowledge that *"Palestinians"* is part of arg_1 and *"converged"* is a verb. If we have the syntactic parse tree of the sentence, we can obtain even more complex and discriminative features. For example, the syntactic parse tree of the same relation instance contains the following subtree: [VP $\rightarrow$ VBD [PP $\rightarrow$ [IN $\rightarrow$ *on*] NP]]. If we know that arg_2 is contained in the NP in this subtree, then this subtree states that arg_2 is in a PP that is attached to a VP, and the proposition is *"on"*. This subtree therefore may also a useful feature. Similarly, if we have the dependency parse tree of the relation instance, then the dependency link *"square $\rightsquigarrow$ on"* states that the token *"square"* is dependent on the token *"on"*, which may also be a useful feature.

Given that useful features are of various forms, in order to systematically search the feature space, we need to first have a unified view of features. This problem is not trivial because it is not immediately clear how different types of features can be unified. We observe, however, that in general features fall into two categories: (1) properties of a single token, and (2) relations between tokens. Features that involve attributes of a single token, such as bag-of-word features and entity attribute features, belong to the first category, while features that involve sequence, syntactic or dependency relations between tokens belong to the second category. Motivated by this observation, we can represent relation instances as graphs, with nodes denoting single tokens or syntactic categories such as NPs and VPs, and edges denoting various types of relations between the nodes.

4.2 Relation Instance Graphs

We represent a relation instance as a labeled, directed graph $G = (V, E, A, B)$, where V is the set of nodes in the graph, E is the set of directed edges in the graph, and A, B are functions that assign labels to the nodes.

First, for each node $v \in V$, $A(v) = \{a_1, a_2, \ldots, a_{|A(v)|}\}$ is a set of attributes associated with node v, where $a_i \in \Sigma$, and Σ is an alphabet that contains all possible attribute values. The attributes are introduced to help generalize the node. For example, if node v represents a token, then $A(v)$ can include the token itself, its morphological base form, its POS, its semantic class (e.g. WordNet synset), etc. If v also happens to be the head word of arg_1 or arg_2, then $A(v)$ can also include the entity type and other entity attributes. If node v represents a syntactic category such as an NP or VP, $A(v)$ can simply contain only the syntactic tag.

Next, function $B : V \rightarrow \{0, 1, 2, 3\}$ is introduced to distinguish argument nodes from non-argument nodes. For each node $v \in V$, $B(v)$ indicates how node v is related to arg_1 and arg_2. 0 indicates that v does not cover any argument, 1 or 2 indicates that v covers arg_1 or arg_2, respectively, and 3 indicates that v covers both arguments. We will see shortly that only nodes that represent syntactic categories in a syntactic parse tree can possibly be assigned 3. We refer to $B(v)$ as the *argument tag* of v.

We now consider three special instantiations of this general definition of relation instance graphs. See Figures 1, 2 and 3 for examples of each of the three representations.

Sequence: Without introducing any additional structured information, a sequence representation preserves the order of the tokens as they occur in the original sentence. Each node in this graph is a token augmented with its relevant attributes. For example, head words of arg_1 and arg_2 are augmented with the corresponding entity types. A token is assigned the argument tag 1 or 2 if it is the head word of arg_1 or

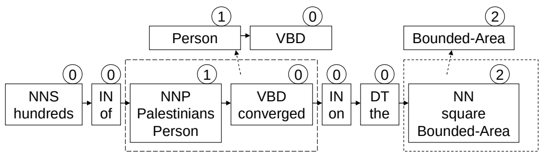


Figure 1: An example sequence representation. The subgraph on the left represents a bigram feature. The subgraph on the right represents a unigram feature that states the entity type of arg_2.

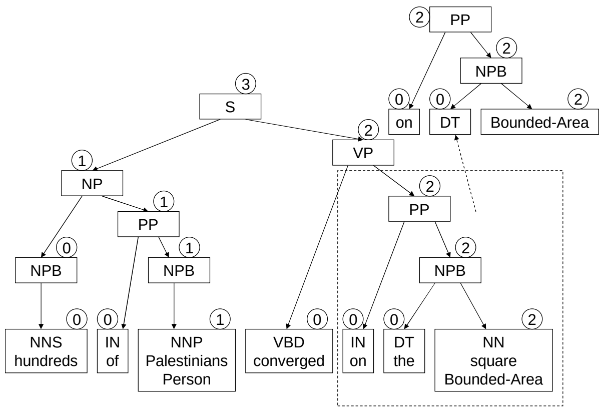


Figure 2: An example syntactic parse tree representation. The subgraph represents a subtree feature (grammar production feature).

arg_2. Otherwise, it is assigned the argument tag 0. There is a directed edge from u to v if and only if the token represented by v immediately follows that represented by u in the sentence.

Syntactic Parse Tree: The syntactic parse tree of the relation instance sentence can be augmented to represent the relation instance. First, we modify the tree slightly by conflating each leaf node in the original parse tree with its parent, which is a preterminal node labeled with a POS tag. Then, each node is augmented with relevant attributes if necessary. Argument tags are assigned to the leaf nodes in the same way as in the sequence representation. For an internal node v, argument tag 1 or 2 is assigned if either arg_1 or arg_2 is inside the subtree rooted at v, and 3 is assigned if both arguments are inside the subtree. Otherwise, 0 is assigned to v.

Dependency Parse Tree: Similarly, the dependency parse tree can also be modified to represent the relation instance. Assignment of attributes and argument tags is the same as for the sequence representation. To simplify the representation, we ignore

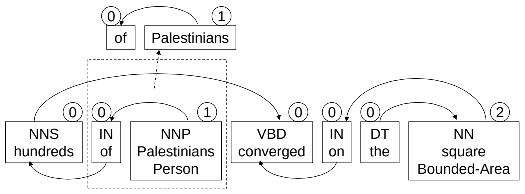


Figure 3: An example dependency parse tree representation. The subgraph represents a dependency relation feature between arg_1 *"Palestinians"* and *"of"*.

the dependency relation types.

4.3 Features

Given the above definition of relation instance graphs, we are now ready to define features. Intuitively, a feature of a relation instance captures part of the attributive and/or structural properties of the relation instance graph. Therefore, it is natural to define a feature as a subgraph of the relation instance graph. Formally, given a graph $G = (V, E, A, B)$, which represents a single relation instance, a feature that exists in this relation instance is a subgraph $G' = (V', E', A', B')$ that satisfies the following conditions: $V' \subseteq V$, $E' \subseteq E$, and $\forall v \in V', A'(v) \subseteq A(v), B'(v) = B(v)$.

We now show that many features that have been explored in previous work on relation extraction can be transformed into this graphic representation. See Figures 1, 2 and 3 for some examples.

Entity Attributes: Previous studies have shown that entity types and entity mention types of arg_1 and arg_2 are very useful (Zhao and Grishman, 2005; Zhou et al., 2005; Zhang et al., 2006b). To represent a single entity attribute, we can take a subgraph that contains only the node representing the head word of the argument, labeled with the entity type or entity mention type. A particularly useful type of features are *conjunctive entity features*, which are conjunctions of two entity attributes, one for each argument. To represent a conjunctive feature such as "arg_1 is a *Person* entity and arg_2 is a *Bounded-Area* entity", we can take a subgraph that contains two nodes, one for each argument, and each labeled with an entity attribute. Note that in this case, the subgraph contains two disconnected components, which is allowed by our definition.

Bag-of-Words: These features have also been explore by Zhao and Grishman (2005) and Zhou et. al. (2005). To represent a bag-of-word feature, we can simply take a subgraph that contains a single node labeled with the token. Because the node also has an argument tag, we can distinguish between argument word and non-argument word.

Bigrams: A bigram feature (Zhao and Grishman, 2005) can be represented by a subgraph consisting of two connected nodes from the sequence representation, where each node is labeled with the token.

Grammar Productions: The features in convolution tree kernels for relation extraction (Zhang et al., 2006a; Zhang et al., 2006b) are sequences of grammar productions, that is, complete subtrees of the syntactic parse tree. Therefore, these features can naturally be represented by subgraphs of the relation instance graphs.

Dependency Relations and Dependency Paths: These features have been explored by Bunescu and Mooney (2005a), Zhao and Grishman (2005), and Zhou et. al. (2005). A dependency relation can be represented as an edge connecting two nodes from the dependency tree. The dependency path between the two arguments can also be easily represented as a path in the dependency tree connecting the two nodes that represent the two arguments.

There are some features that are not covered by our current definition, but can be included if we modify our relation instance graphs. For example, gapped subsequence features in subsequence kernels (Bunescu and Mooney, 2005b) can be represented as subgraphs of the sequence representation if we add more edges to connect any pair of nodes u and v provided that the token represented by u occurs somewhere before that represented by v in the sentence. Since our feature definition is very general, our feature space also includes many features that have not been explored before.

4.4 Searching the Feature Space

Although the feature space we have defined is relatively complete and has a clear structure, it is still too expensive to exhaustively search the space because the number of features is exponential in terms of the size of the relation instance graph. We thus propose to search the feature space in the following bottom-up manner: We start with the conjunctive entity features (defined in Section 4.3), which have been found effective in previous studies and are intuitively necessary for relation extraction. We then systematically add unit features with different granularities. We first consider the minimum (i.e. most basic) unit features. We then gradually include more complex features. The motivations for this strategy are the following: (1) Using the smallest features to represent a relation instance graph presumably covers all unit characteristics of the graph. (2) Using small subgraphs allows fuzzy matching, which is good for our task because relation instances of the same type may vary in their relation instance graphs, especially with the noise introduced by adjectives, adverbs, or irrelevant propositional phrases. (3) The number of features of a fixed small size is polynomial in terms of the size of the relation instance graph. It is therefore feasible to generate all the small unit features and use any classifier such as a maximum entropy classifier or an SVM.

In our experiments, we consider three levels of small unit features in increasing order of their complexity. First, we consider *unigram* features $G_{uni} = (\{u\}, \emptyset, A_{uni}, B)$, where $A_{uni}(u) = \{a_i\} \subseteq A(u)$. In another word, unigram features consist of a single node labeled with a single attribute. Examples of unigram features include bag-of-word features and non-conjunctive entity attribute features. At the second level, we consider *bigram* features $G_{bi} = (\{u, v\}, \{(u, v)\}, A_{uni}, B)$. *Bigram* features are therefore single edges connecting two nodes, where each node is labeled with a single attribute. The third level of attributes we consider are *trigram* features $G_{tri} = (\{u, v, w\}, \{(u, v), (u, w)\}, A_{uni}, B)$ or $G_{tri} = (\{u, v, w\}, \{(u, v), (v, w)\}, A_{uni}, B)$. Thus *trigram* features consist of two connected edges and three nodes, where each node is also labeled with a single attribute.

We treat the three relation instance graphs (sequences, syntactic parse trees, and dependency parse trees) as three feature subspaces, and search in each subspace. For each feature subspace, we incrementally add the unigram, bigram and trigram features to the working feature set. For the syntactic parse tree representation, we also consider a fourth level of small unit features, which are single grammar productions such as [VP $\rightarrow$ VBD PP], because these are the smallest features in convolution tree kernels. After we explore each feature subspace, we try to combine the features from the three subspaces to see whether the performance can be improved, that is, we test whether the sequence, syntactic and dependency relations can complement each other.

5 Experiments

5.1 Data Set and Experiment Setup

We used the data set from ACE (Automatic Content Extraction) 2004 evaluation to conduct our experiments. This corpus defines 7 types of relations: *Physical*, *Personal / Social*, *Empolyment / Memebership / Subsidiary*, *Agent-Artifact*, *PER / ORG Affiliation*, *GPE Affiliation* and *Discourse*.

We used Collins parser to parse the sentences in the corpus because Collins parser gives us the head of each syntactic category, which allows us to transform the syntactic parse trees into dependency trees. We discarded sentences that could not be parsed by Collins parser. The candidate relation instances were generated by considering all pairs of entities that occur in the same sentence. We obtained 48625 candidate relation instances in total, among which 4296 instances were positive.

As in most existing work, instead of using the entire sentence, we used only the sequence of tokens that are inside the minimum complete subtree covering the two arguments. Presumably, tokens outside of this subtree are not so relevant to the task. In our graphic representation of relation instances, the attribute set for a token node includes the token itself, its POS tag, and entity type, entity subtype and entity mention type when applicable. The attribute set for a syntactic category node includes only the syntactic tag. We used both maximum entropy classifier and SVM for all experiments. We adopted one vs. others strategy for the multi-class classification problem. In all experiments, the performance shown was based on 5-fold cross validation.

5.2 General Search in the Feature Subspaces

Following the general search strategy, we conducted the following experiments. For each feature subspace, we started with the conjunctive entity features plus the unigram features. We then incrementally added bigram and trigram features. For the syntactic parse tree feature space, we conducted an additional experiment: We added basic grammar production features on top of the unigram, bigram and trigram features. Adding production features allows us to study the effect of adding more complex and presumably more specific and discriminative features.

Table 1 shows the precision (P), recall (R) and F1 measure (F) from the experiments with the maximum entropy classifier (ME) and the SVM classifier (SVM). We can compare the results in two dimensions. First, within each feature subspace, while bigram features improved the performance significantly over unigrams, trigrams did not improve the performance very much. This trend is observed for both classifiers. In the case of the syntactic parse tree subspace, adding production features even hurt the performance. This suggests that inclusion of complex features is not guaranteed to improve the performance.

Second, if we compare the best performance achieved in each feature subspace, we can see that for both classifiers, syntactic parse tree is the most effective feature space, while sequence and dependency tree are similar. However, the difference in performance between the syntactic parse tree subspace and the other two subspaces is not very large. This suggests that each feature subspace alone already captures most of the useful structural information between tokens for relation extraction. The reason why the sequence feature subspace gave good performance although it contained the least structural information is probably that many relations defined in the ACE corpus are short-range relations, some within single noun phrases. For such kind of relations, sequence information may be even more reliable than syntactic or dependency information, which may not be accurate due to parsing errors.

Next, we conducted experiments to combine the features from the three subspaces to see whether this could further improve the performance. For sequence subspace and dependency tree subspace, we used up to bigram features, and for syntactic parse tree subspace, we used up to trigram features. In Table 2, we show the experiment results. We can see that for both classifiers, adding features from the sequence subspace or from the dependency tree subspace to the syntactic parse tree subspace can improve the performance slightly. But combining sequence subspace and dependency tree subspace does not generate any performance improvement. Again,

			Uni	+Bi	+Tri	+Prod
ME	Seq	P	0.647	0.662	0.717	N/A
		R	0.614	0.701	0.653	
		F	0.630	0.681	0.683	
	Syn	P	0.651	0.695	0.726	0.702
		R	0.645	0.698	0.688	0.691
		F	0.648	0.697	**0.707**	0.696
	Dep	P	0.647	0.673	0.718	N/A
		R	0.614	0.676	0.652	
		F	0.630	0.674	0.683	
SVM	Seq	P	0.583	0.666	0.684	N/A
		R	0.586	0.650	0.648	
		F	0.585	0.658	0.665	
	Syn	P	0.598	0.645	0.679	0.674
		R	0.611	0.663	0.681	0.672
		F	0.604	0.654	**0.680**	0.673
	Dep	P	0.583	0.644	0.682	N/A
		R	0.586	0.638	0.645	
		F	0.585	0.641	0.663	

Table 1: Comparison among the three feature subspaces and the effect of including larger features.

		Seq+Syn	Seq+Dep	Syn+Dep	All
ME	P	0.737	0.687	0.695	0.724
	R	0.694	0.682	0.731	0.702
	F	**0.715**	0.684	0.712	0.713
SVM	P	0.689	0.669	0.687	0.691
	R	0.686	0.653	0.682	0.686
	F	**0.688**	0.661	0.684	**0.688**

Table 2: The effect of combining the three feature subspaces.

this suggests that since many of the ACE relations are local, there is likely much overlap between sequence information and dependency information.

We also tried the convolution tree kernel method (Zhang et al., 2006a), using an SVM tree kernel package[2]. The performance we obtained was P = 0.705, R = 0.685, and F = 0.695[3]. This F measure is higher than the best SVM performance in Table 1. The convolution tree kernel uses large subtree features, but such features are deemphasized with an exponentially decaying weight. We found that the performance was sensitive to this decaying factor, suggesting that complex features can be useful if they are weighted appropriately, and further study of how to optimize the weights of such complex features is needed.

[2]http://ai-nlp.info.uniroma2.it/moschitti/Tree-Kernel.htm

[3]The performance we achieved is lower than that reported in (Zhang et al., 2006b), due to different data preprocessing, data partition, and parameter setting.

5.3 Task-Oriented Feature Pruning

Apart from the general bottom-up search strategy we have proposed, we can also introduce some task-oriented heuristics based on intuition or domain knowledge to prune the feature space. In our experiments, we tried the following heuristics.

H1: Zhang et al. (2006a) found that using *path-enclosed tree* performed better than using *minimum complete tree*, when convolution tree kernels were applied. In path-enclosed trees, tokens before arg_1 and after arg_2 as well as their links with other nodes in the tree are removed. Based on this previous finding, our first heuristic was to change the syntactic parse tree representation of the relation instances into path-enclosed trees.

H2: We hypothesize that words such as articles, adjectives and adverbs are not very useful for relation extraction. We thus removed sequence unigram features and bigram features that contain an article, adjective or adverb.

H3: Similar to H2, we can remove bigrams in the syntactic parse tree subspace if the bigram contains an article, adjective or adverb.

H4: Similar to H1, we can also remove the tokens before arg_1 and after arg_2 from the sequence representation of a relation instance.

In Table 3, we show the performance after applying these heuristics. We started with the best configuration from our previous experiments, that is, combing up to bigram features in the sequence subspace and up to trigram features in the syntactic tree subspace. We then applied heuristics H1 to H4 incrementally unless we saw that a heuristic was not effective. We found that H1, H2 and H4 slightly improved the performance, but H3 hurt the performance. On the one hand, the improvement suggests that our original feature configuration included some irrelevant features, and in turn confirmed that over-inclusion of features could hurt the performance. On the other hand, since the improvement brought by H1, H2 and H4 was rather small, and H3 even hurt the performance, we could see that it is in general very hard to find good feature pruning heuristics.

6 Conclusions and Future Work

In this paper, we conducted a systematic study of the feature space for relation extraction. We pro-

	ME			SVM		
	P	R	F	P	R	F
Best	0.737	0.694	0.715	0.689	0.686	0.688
+H1	0.714	0.729	0.721	0.698	0.699	0.699
+H2	0.730	0.723	0.726	0.704	0.704	**0.704**
+H3	0.739	0.704	0.721	0.701	0.696	0.698
-H3+H4	0.746	0.713	**0.729**	0.702	0.701	0.702

Table 3: The effect of various heuristic feature pruning methods.

posed and defined a unified graphic representation of features for relation extraction, which serves as a general framework for systematically exploring features defined on natural language sentences. With this framework, we explored three different representations of sentences—sequences, syntactic parse trees, and dependency trees—which lead to three feature subspaces. In each subspace, starting with the basic unit features, we systematically explored features of different levels of complexity. The studied feature space includes not only most of the effective features explored in previous work, but also some features that have not been considered before.

Our experiment results showed that using a set of basic unit features from each feature subspace, we can achieve reasonably good performance. When the three subspaces are combined, the performance can improve only slightly, which suggests that the sequence, syntactic and dependency relations have much overlap for the task of relation extraction. We also found that adding more complex features may not improve the performance much, and may even hurt the performance. A combination of features of different levels of complexity and from different sentence representations, coupled with task-oriented feature pruning, gives the best performance.

In our future work, we will study how to automatically conduct task-oriented feature search, feature pruning and feature weighting using statistical methods instead of heuristics. In this study, we only considered features from the local context, i.e. the sentence that contains the two arguments. Some existing studies use corpus-based statistics for relation extraction (Hasegawa et al., 2004). In the future, we will study the effectiveness of these global features.

Acknowledgments

This work was in part supported by the National Science Foundation under award numbers 0425852 and 0428472. We thank Alessandro Moschitti for providing the SVM tree kernel package. We also thank Min Zhang for providing the implementation details of the convolution tree kernel for relation extraction.

References

Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kenrel for relation extraction. In *Proceedings of HLT/EMNLP*.

Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings of NIPS*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*.

Chad Cumby and Dan Roth. 2003. On kernel methods for relational learning. In *Proceedings of ICML*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings ACL*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL*.

Jun Suzuki, Tsutomu Hirao, Yutaka Sasaki, and Eisaku Maeda. 2003. Hierarchical directed acyclic graph kernel: Methods for structured natural language data. In *Proceedings of ACL*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Min Zhang, Jie Zhang, and Jian Su. 2006a. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of HLT/NAACL*.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006b. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of ACL*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of ACL*.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL*.

Unsupervised Resolution of Objects and Relations on the Web

Alexander Yates
Turing Center
Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
ayates@cs.washington.edu

Oren Etzioni
Turing Center
Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
etzioni@cs.washington.edu

Abstract

The task of identifying synonymous relations and objects, or Synonym Resolution (SR), is critical for high-quality information extraction. The bulk of previous SR work assumed strong domain knowledge or hand-tagged training examples. This paper investigates SR in the context of unsupervised information extraction, where neither is available. The paper presents a scalable, fully-implemented system for SR that runs in $O(KN\ log\ N)$ time in the number of extractions N and the maximum number of synonyms per word, K. The system, called RESOLVER, introduces a probabilistic relational model for predicting whether two strings are co-referential based on the similarity of the assertions containing them. Given two million assertions extracted from the Web, RESOLVER resolves objects with 78% precision and an estimated 68% recall and resolves relations with 90% precision and 35% recall.

1 Introduction

Web Information Extraction (WIE) systems extract *assertions* that describe a relation and its arguments from Web text (e.g., $(\text{is capital of}, \text{D.C.}, \text{United States})$). WIE systems can extract hundreds of millions of assertions containing millions of different strings from the Web (*e.g.*, the TEXTRUNNER system (Banko et al., 2007)).[1] WIE systems often extract assertions that describe the same real-world object or relation using different names. For example, a WIE system might extract $(\text{is capital city of}, \text{Washington}, \text{U.S.})$, which describes the same relationship as above but contains a different name for the relation and each argument.

Synonyms are prevalent in text, and the Web corpus is no exception. Our data set of two million assertions extracted from a Web crawl contained over a half-dozen different names each for the United States and Washington, D.C., and three for the "is capital of" relation. The top 80 most commonly extracted objects had an average of 2.9 extracted names per entity, and several had as many as 10 names. The top 100 most commonly extracted relations had an average of 4.9 synonyms per relation.

We refer to the problem of identifying synonymous object and relation names as Synonym Resolution (SR).[2] An SR system for WIE takes a set of assertions as input and returns a set of clusters, with each cluster containing coreferential object strings or relation strings. Previous techniques for SR have focused on one particular aspect of the problem, either objects or relations. In addition, the techniques either depend on a large set of training examples, or are tailored to a specific domain by assuming knowledge of the domain's schema. Due to the number and diversity of the relations extracted, these tech-

[1] For a demo see www.cs.washington.edu/research/textrunner.

[2] Ironically, SR has a number of synonyms in the literature, including Entity Resolution, Record Linkage, and Deduplication.

niques are not feasible for WIE systems. Schemata are not available for the Web, and hand-labeling training examples for each relation would require a prohibitive manual effort.

In response, we present RESOLVER, a novel, domain-independent, unsupervised synonym resolution system that applies to both objects and relations. RESOLVER clusters coreferential names together using a probabilistic model informed by string similarity and the similarity of the assertions containing the names. Our contributions are:

1. A scalable clustering algorithm that runs in time $O(KN\ log\ N)$ in the number of extractions N and maximum number of synonyms per word, K, without discarding any potentially matching pair, under exceptionally weak assumptions about the data.
2. An unsupervised probabilistic model for predicting whether two object or relation names co-refer.
3. An empirical demonstration that RESOLVER can resolve objects with 78% precision and 68% recall, and relations with 90% precision and 35% recall.

The next section discusses previous work. Section 3 introduces our probabilistic model for SR. Section 4 describes our clustering algorithm. Section 5 describes extensions to our basic SR system. Section 6 presents our experiments, and section 7 discusses our conclusions and areas for future work.

2 Previous Work

The DIRT algorithm (Lin and Pantel, 2001) addresses a piece of the unsupervised SR problem. DIRT is a heuristic method for finding synonymous relations, or "inference rules." DIRT uses a dependency parser and mutual information statistics over a corpus to identify relations that have similar sets of arguments. In contrast, our algorithm provides a formal probabilistic model that applies equally well to relations and objects, and we provide an evaluation of the algorithm in terms of precision and recall.

There are many unsupervised approaches for object resolution in databases, but unlike our algorithm these approaches depend on a known, fixed schema. Ravikumar and Cohen (Ravikumar and Cohen, 2004) present an unsupervised approach to object resolution using Expectation-Maximization on a hierarchical graphical model. Several other recent approaches leverage domain-specific information and heuristics for object resolution. For example, many (Dong et al., 2005; Bhattacharya and Getoor, 2005; Bhattacharya and Getoor, 2006) rely on evidence from observing which strings appear as arguments to the same relation simultaneously (*e.g.*, co-authors of the same publication). While this is useful information when resolving authors in the citation domain, it is extremely rare to find relations with similar properties in extracted assertions. None of these approaches applies to the problem of resolving relations. See (Winkler, 1999) for a survey of this area.

Several supervised learning techniques make entity resolution decisions (Kehler, 1997; McCallum and Wellner, 2004; Singla and Domingos, 2006), but of course these systems depend on the availability of training data, and often on a significant number of labeled examples per relation of interest. These approaches also depend on complex probabilistic models and learning algorithms, and they have order $O(n^3)$ time complexity, or worse. They currently do not scale to the amounts of data extracted from the Web. Previous systems were tested on at most a few thousand examples, compared with millions or hundreds of millions of extractions from WIE systems such as TEXTRUNNER.

Coreference resolution systems (e.g., (Lappin and Leass, 1994; Ng and Cardie, 2002)), like SR systems, try to merge references to the same object (typically pronouns, but potentially other types of noun phrases). This problem differs from the SR problem in several ways: first, it deals with unstructered text input, possibly with syntactic annotation, rather than relational input. Second, it deals only with resolving objects. Finally, it requires local decisions about strings; that is, the same word may appear twice in a text and refer to two different things, so each occurrence of a word must be treated separately.

The PASCAL Recognising Textual Entailment Challenge proposes the task of recognizing when two sentences entail one another, and many authors have submitted responses to this challenge (Dagan et al., 2006). Synonym resolution is a subtask of this problem. Our task differs significantly from the textual entailment task in that it has no labeled training data, and its input is in the form of relational extractions rather than raw text.

Two probabilistic models for information extraction have a connection with ours. Our probabilistic model is partly inspired by the ball-and-urns abstraction of information extraction presented by Downey et al. (2005) Our task and probability model are different from theirs, but we make many of the same modeling assumptions. Second, we follow Snow et al.'s work (2006) on taxonomy induction in incorporating transitive closure constraints in our probability calculations, as explained below.

3 Probabilistic Model

Our probabilistic model provides a formal, rigorous method for resolving synonyms in the absence of training data. It has two sources of evidence: the similarity of the strings themselves (i.e., edit distance) and the similarity of the assertions they appear in. This second source of evidence is sometimes referred to as "distributional similarity" (Hindle, 1990).

Section 3.2 presents a simple model for predicting whether a pair of strings co-refer based on string similarity. Section 3.3 then presents a model called the Extracted Shared Property (ESP) Model for predicting whether a pair of strings co-refer based on their distributional similarity. Finally, a method is presented for combining these models to come up with an overall prediction for coreference decisions between two clusters of strings.

3.1 Terminology and Notation

We use the following notation to describe the probabilistic models. The input is a data set D containing extracted *assertions* of the form $a = (r, o_1, \ldots, o_n)$, where r is a relation string and each o_i is an object string representing the arguments to the relation. In our data, all of the extracted assertions are binary, so $n = 2$. The subset of all assertions in D containing a string s is called D_s.

For strings s_i and s_j, let $R_{i,j}$ be the random variable for the event that s_i and s_j refer to the same entity. Let $R^t_{i,j}$ denote the event that $R_{i,j}$ is true, and $R^f_{i,j}$ denote the event that it is false.

A pair of strings (r, s_2) is called a *property* of a string s_1 if there is an assertion $(r, s_1, s_2) \in D$ or $(r, s_2, s_1) \in D$. A pair of strings (s_1, s_2) is an *instance* of a string r if there is an assertion $(r, s_1, s_2) \in D$. Equivalently, the property $p = (r, s_2)$ *applies* to s_1, and the relation r *applies* to the instance $i = (s_1, s_2)$. Finally, two strings x and y *share* a property (or instance) if both x and y are extracted with the same property (or instance).

3.2 String Similarity Model

Many objects appear with multiple names that are substrings, acronyms, abbreviations, or other simple variations of one another. Thus string similarity can be an important source of evidence for whether two strings co-refer. Our probabilistic String Similarity Model (SSM) assumes a similarity function $\text{sim}(s_1, s_2)$: $STRING \times STRING \rightarrow [0, 1]$. The model sets the probability of s_1 co-referring with s_2 to a smoothed version of the similarity:

$$P(R^t_{i,j} | \text{sim}(s_1, s_2)) = \frac{\alpha * \text{sim}(s_1, s_2) + 1}{\alpha + \beta}$$

The particular choice of α and β make little difference to our results, so long as they are chosen such that the resulting probability can never be one or zero. In our experiments $\alpha = 20$ and $\beta = 5$, and we use the well-known Monge-Elkan string similarity function for objects and the Levenshtein string edit-distance function for relations (Cohen et al., 2003).

3.3 The Extracted Shared Property Model

The Extracted Shared Property (ESP) Model outputs the probability that s_1 and s_2 co-refer based on how many properties (or instances) they share. As an example, consider the strings "Mars" and "Red Planet", which appear in our data 659 and 26 times respectively. Out of these extracted assertions, they share four properties. For example, $(lacks, Mars, ozone\ layer)$ and $(lacks, Red\ Planet, ozone\ layer)$ both appear as assertions in our data. The ESP model determines the probability that "Mars" and "Red Planet" refer to the same entity after observing k, the number of properties that apply to both, n_1, the total number of extracted properties for "Mars", and n_2, the total number of extracted properties for "Red Planet."

ESP models the extraction of assertions as a generative process, much like the URNS model (Downey et al., 2005). For each string s_i, a certain number, P_i, of properties of the string are written on balls and placed in an urn. Extracting n_i assertions that contain s_i amounts to selecting a subset of size n_i from these labeled balls.[3] Properties in the urn are called *potential properties* to distinguish them from extracted properties.

To model coreference decisions, ESP uses a pair of urns, containing P_i and P_j balls respectively, for the two strings s_i and s_j. Some subset of the P_i balls have the exact same labels as an equal-sized subset of the P_j balls. Let the size of this subset be $S_{i,j}$. The ESP model assumes that coreferential strings share as many potential properties as possible, though only a few of the potential properties will be extracted for both. For non-coreferential strings, the number of shared potential properties is a strict subset of the potential properties of each string. Thus if $R_{i,j}$ is true then $S_{i,j} = \min(P_i, P_j)$, and if $R_{i,j}$ is false then $S_{i,j} < \min(P_i, P_j)$.

The ESP model makes several simplifying assumptions in order to make probability predictions. As is suggested by the ball-and-urn abstraction, it assumes that each ball for a string is equally likely to be selected from its urn. Because of data sparsity, almost all properties are very rare, so it would be difficult to get a better estimate for the prior probability of selecting a particular potential property. Second, it assumes that without knowing the value of k, every value of $S_{i,j}$ is equally likely, since we have no better information. Finally, it assumes that all subsets of potential properties are equally likely to be shared by two non-coreferential objects, regardless of the particular labels on the balls, given the size of the shared subset.

Given these assumptions, we can derive an expression for $P(R^t_{i,j})$. First, note that there are $\binom{P_i}{n_i}\binom{P_j}{n_j}$ total ways of extracting n_i and n_j assertions for s_i and s_j. Given a particular value of $S_{i,j}$, the number of ways in which n_i and n_j assertions can be extracted such that they share exactly k is given by

$$\mathrm{Count}(k, n_i, n_j | P_i, P_j, S_{i,j}) =$$

$$\binom{S_{i,j}}{k} \sum_{r,s\geq 0} \binom{S_{i,j}-k}{r+s}\binom{r+s}{r}\binom{P_i - S_{i,j}}{n_i-(k+r)}\binom{P_j - S_{i,j}}{n_j-(k+s)}$$

By our assumptions,

$$P(k|n_i, n_j, P_i, P_j, S_{i,j}) =$$

$$\frac{\mathrm{Count}(k, n_i, n_j | P_i, P_j, S_{i,j})}{\binom{P_i}{n_i}\binom{P_j}{n_j}} \qquad (1)$$

Let $P_{\min} = \min(P_i, P_j)$. The result below follows from Bayes' Rule and our assumptions above:

Proposition 1 *If two strings s_i and s_j have P_i and P_j potential properties (or instances), and they appear in extracted assertions D_i and D_j such that $|D_i| = n_i$ and $|D_j| = n_j$, and they share k extracted properties (or instances), the probability that s_i and s_j co-refer is:*

$$P(R^t_{i,j}|D_i, D_j, P_i, P_j) =$$

$$\frac{P(k|n_i, n_j, P_i, P_j, S_{i,j} = P_{\min})}{\sum_{k\leq S_{i,j}\leq P_{\min}} P(k|n_i, n_j, P_i, P_j, S_{i,j})} \qquad (2)$$

Substituting equation 1 into equation 2 gives us a complete expression for the probability we are looking for.

Note that the probability for $R_{i,j}$ depends on just two hidden parameters, P_i and P_j. Since we have no labeled data to estimate these parameters from, we tie these parameters to the number of times the respective strings s_i and s_j are extracted. Thus we set $P_i = N \times n_i$, and we set $N = 50$ in our experiments.

3.4 Combining the Evidence

For each potential coreference relationship $R_{i,j}$, there are now two pieces of probabilistic evidence. Let $E^e_{i,j}$ be the evidence for ESP, and let $E^s_{i,j}$ be the evidence for SSM. Our method for combining the two uses the Naïve Bayes assumption that each piece of evidence is conditionally independent, given the coreference relation:

$$P(E^s_{i,j}, E^e_{i,j}|R_{i,j}) = P(E^s_{i,j}|R_{i,j})P(E^e_{i,j}|R_{i,j})$$

Given this simplifying assumption, we can combine the evidence to find the probability of a coference relationship by applying Bayes' Rule to both sides (we omit the i, j indices for brevity):

$$P(R^t|E^s, E^e) =$$

$$\frac{P(R^t|E^s)P(R^t|E^e)(1 - P(R^t))}{\sum_{i\in\{t,f\}} P(R^i|E^s)P(R^i|E^e)(1 - P(R^i))}$$

[3]Unlike the URNS model, balls are drawn without replacement because each extracted property is distinct in our data.

3.5 Comparing Clusters of Strings

Our algorithm merges clusters of strings with one another, using one of the above models. However, these models give probabilities for coreference decisions between two individual strings, not two clusters of strings.

We follow the work of Snow et al. (2006) in incorporating transitive closure constraints in probabilistic modeling, and make the same independence assumptions. The benefit of this approach is that the calculation for merging two clusters depends only on coreference decisions between individual strings, which can be calculated independently.

Let a *clustering* be a set of coreference relationships between pairs of strings such that the coreference relationships obey the transitive closure property. We let the probability of a set of assertions D given a clustering C be:

$$\begin{aligned} P(D|C) = & \prod_{R^t_{i,j} \in C} P(D_i \cup D_j | R^t_{i,j}) \times \\ & \prod_{R^f_{i,j} \in C} P(D_i \cup D_j | R^f_{i,j}) \end{aligned}$$

The metric used to determine if two clusters should be merged is the likelihood ratio, or the probability for the set of assertions given the merged clusters over the probability given the original clustering. Let C' be a clustering that differs from C only in that two clusters in C have been merged in C', and let ΔC be the set of coreference relationships in C' that are true, but the corresponding ones in C are false. This metric is given by:

$$P(D|C')/P(D|C) =$$

$$\frac{\prod_{R^t_{i,j} \in \Delta C} P(R^t_{i,j}|D_i \cup D_j)(1 - P(R^t_{i,j}))}{\prod_{R^t_{i,j} \in \Delta C} (1 - P(R^t_{i,j}|D_i \cup D_j))P(R^t_{i,j})}$$

The probability $P(R^t_{i,j}|D_i \cup D_j)$ may be supplied by the SSM, ESP, or combination model. In our experiments, we let the prior for the SSM model be 0.5. For the ESP and combined models, we set the prior to $P(R^t_{i,j}) = \frac{1}{\min(P_1, P_2)}$.

4 RESOLVER's Clustering Algorithm

Our clustering algorithm iteratively merges clusters of co-referential names, making each iteration in time $O(N\ log\ N)$ in the number of extracted assertions. The algorithm requires only basic assumptions about which strings to compare. Previous work on speeding up clustering algorithms for SR has either required far stronger assumptions, or else it has focused on heuristic methods that remain, in the worst case, $O(N^2)$ in the number of distinct objects.

Our algorithm, a greedy agglomerative clustering method, is outlined in Figure 1. The first novel part of the algorithm, step 3, compares pairs of strings that share the same property or instance, so long as no more than Max strings share that same property or instance. After the scores for all comparisons are made, each string is assigned its own cluster. Then the scores are sorted and the best cluster pairs are merged until no pair of clusters has a score above threshold. The second novel aspect of this algorithm is that as it merges clusters in Step 4, it merges properties containing those clusters in a process we call *mutual recursion*, which is discussed below.

This algorithm compares every pair of clusters that have the potential to be merged, assuming two properties of the data. First, it assumes that pairs of clusters with no shared properties are not worth

```
$S$ := set of all strings
For each property or instance $p$,
    $S_p := \{s \in S | s$ has property $p\}$
1. $Scores := \{\}$
2. Build index mapping properties (and instances)
    to strings with those properties (instances)
3. For each property or instance $p$:
    If $|S_p| < \text{Max}$:
        For each pair $\{s1, s2\} \subset S_p$:
            Add mergeScore$(s1, s2)$ to $Scores$
4. Repeat until no merges can be performed:
    Sort $Scores$
    $UsedClusters := \{\}$
    While score of top clusters $c_1, c_2$
      is above Threshold:
        Skip if either is in $UsedClusters$
        Merge $c_1$ and $c_2$
        Add $c_1, c_2$ to $UsedClusters$
        Merge properties containing $c_1, c_2$
    Recalculate merge scores as in Steps 1-3
```

Figure 1: RESOLVER's Clustering Algorithm

comparing. Since the number of shared properties is a key source of evidence for our approach, these clusters almost certainly will not be merged, even if they are compared, so the assumption is quite reasonable. Second, the approach assumes that clusters sharing only properties that apply to very many strings (more than Max) need not be compared. Since properties shared by many strings provide little evidence that the strings are coreferential, this assumption is reasonable for SR. We use $Max = 50$ in our experiments. Less than 0.1% of the properties are thrown out using this cutoff.

4.1 Algorithm Analysis

Let D be the set of extracted assertions. The following analysis shows that one iteration of merges takes time $O(N\ log\ N)$, where $N = |D|$. Let NC be the number of comparisons between strings in step 3. To simplify the analysis, we consider only those properties that contain a relation string and an argument 1 string. Let A be the set of all such properties. NC is linear in N:[4]

$$\begin{aligned} NC &= \sum_{p \in A} \frac{|S_p| \times (|S_p| - 1)}{2} \\ &\leq \frac{(Max - 1)}{2} \times \sum_{p \in A} |S_p| \\ &= \frac{(Max - 1)}{2} \times N \end{aligned}$$

Note that this bound is quite loose because most properties apply to only a few strings. Step 4 requires time $O(N\ log\ N)$ to sort the comparison scores and perform one iteration of merges. If the largest cluster has size K, in the worst case the algorithm will take K iterations. In our experiments, the algorithm never took more than 9 iterations.

4.2 Relation to other speed-up techniques

The merge/purge algorithm (Hernandez and Stolfo, 1995) assumes the existence of a particular attribute such that when the data set is sorted on this attribute, matching pairs will all appear within a narrow window of one another. This algorithm is $O(M\ log\ M)$ where M is the number of distinct strings. However, there is no attribute or set of attributes that comes close to satisfying this assumption in the context of domain-independent information extraction.

There are several techniques that often provide speed-ups in practice, but in the worst case they make $O(M^2)$ comparisons at each merge iteration, where M is the number of distinct strings. This can cause problems on very large data sets. Notably, McCallum et al. (2000) use a cheap comparison metric to place objects into overlapping "canopies," and then use a more expensive metric to cluster objects appearing in the same canopy. The RESOLVER clustering algorithm is in fact an adaptation of the canopy method; it adds the restriction that strings are not compared when they share only high-frequency properties. The canopy method works well on high-dimensional data with many clusters, which is the case with our problem, but its time complexity is worse than ours.

For information extraction data, a complexity of $O(M^2)$ in the number of distinct strings turns out to be considerably worse than our algorithm's complexity of $O(N\ log\ N)$ in the number of extracted assertions. This is because the data obeys a Zipf law relationship between the frequency of a string and its rank, so the number of distinct strings grows linearly or almost linearly with the number of assertions.[5]

4.3 Mutual Recursion

Mutual recursion refers to the novel property of our algorithm that as it clusters relation strings together into sets of synonyms, it collapses properties together for object strings and potentially finds more shared properties between coreferential object strings. Likewise, as it clusters objects together into sets of coreferential names, it collapses instances of relations together and potentially finds more shared instances between coreferential relations. Thus the clustering decisions for relations and objects mutually depend on one another.

For example, the strings "Kennedy" and "President Kennedy" appear in 430 and 97 assertions in our data, respectively, but none of their extracted properties match exactly. Many properties,

[4]If the Max parameter is allowed to vary with $log|D|$, rather than remaining constant, the same analysis leads to a slightly looser bound that is still better than $O(N^2)$.

[5]The exact relationship depends on the shape parameter z of the Zipf curve. If $z < 1$, as it is for our data set, the number of total extractions grows linearly with the number of distinct strings extracted. If $z = 1$, then n extractions will contain $\Omega(\frac{n}{\ln n})$ distinct strings.

however, *almost* match. For example, the assertions $\langle \text{challenged}, \text{Kennedy}, \text{Premier Krushchev}\rangle$ and $\langle \text{stood up to}, \text{President Kennedy}, \text{Kruschev}\rangle$ both appear in our data. Because "challenged" and "stood up to" are similar, and "Krushchev" and "Premier Krushchev" are similar, our algorithm is able to merge these pairs into two clusters, thereby creating a new shared property between "Kennedy" and "President Kennedy." Eventually it can merge these two strings as well.

5 Extensions to RESOLVER

While the basic RESOLVER system can cluster synonyms accurately and quickly, there is one type of error that it frequently makes. In some cases, it has difficulty distinguishing between similar pairs of objects and *identical* pairs. For example, "Virginia" and "West Virginia" share several extractions because they have the same type, and they have high string similarity. As a result, RESOLVER clusters these two together. The next two sections describe two extensions to RESOLVER that address the problem of similarity vs. identity.

5.1 Function Filtering

RESOLVER can use functions and one-to-one relations to help distinguish between similar and identical pairs. For example, West Virginia and Virginia have different capitals: Richmond and Charleston, respectively. If both of these facts are extracted, and if RESOLVER knows that the "capital of" relation is functional, it should prevent Virginia and West Virginia from merging.

The Function Filter prevents merges between strings that have different values for the same function. More precisely, it decides that two strings y_1 and y_2 *match* if their string similarity is above a high threshold. It prevents a merge between strings x_1 and x_2 if there exist a function f and extractions $f(x_1, y_1)$ and $f(x_2, y_2)$, and there are no such extractions such that y_1 and y_2 match (and *vice versa* for one-to-one relations). Experiments described in section 6 show that the Function Filter can improve the precision of RESOLVER without significantly affecting its recall.

While the Function Filter currently uses functions and one-to-one relations as negative evidence, it is also possible to use them as positive evidence. For example, the relation "married" is not strictly one-to-one, but for most people the set of spouses is very small. If a pair of strings are extracted with the same spouse—*e.g.*, "FDR" and "President Roosevelt" share the property ("married", "Eleanor Roosevelt")—this is far stronger evidence that the two strings are identical than if they shared some random property.

Unfortunately, various techniques that attempted to model this insight, including a TF-IDF weighting of properties, yielded essentially no improvement of RESOLVER. One major reason is that there are relatively few examples of shared functional or one-to-one properties because of sparsity. This idea deserves more investigation, however, and is an area for future work.

5.2 Using Web Hitcounts

While names for two similar objects may often appear together in the same sentence, it is relatively rare for two different names of the same object to appear in the same sentence. RESOLVER exploits this fact by querying the Web to determine how often a pair of strings appears together in a large corpus. When the hitcount is high, RESOLVER can prevent the merge.

Specifically, the Coordination-Phrase Filter searches for hitcounts of the phrase "s_1 and s_2", where s_1 and s_2 are a candidate pair for merging. It then computes a variant of pointwise mutual information, given by

$$\text{coordination score}(s_1, s_2) = \frac{\text{hits}(s_1 \text{ and } s_2)^2}{\text{hits}(s_1) \times \text{hits}(s_2)}$$

The filter prevents any merge for which the coordination score is above a threshold, which is determined on a development set. The results of Coordination-Phrase filtering are discussed in the next section.

6 Experiments

Our experiments demonstrate that the ESP model is significantly better at resolving synonyms than a widely-used distributional similarity metric, the cosine similarity metric (CSM) (Salton and McGill, 1983), and that RESOLVER is significantly better at resolving synonyms than either of its components, SSM or ESP.

We test these models on a data set of 2.1 million assertions extracted from a Web crawl.[6] All models ran over all assertions, but compared only those objects or relations that appeared at least 25 times in the data, to give the ESP and CSM models sufficient data for estimating similarity. However, the models do use strings that appear less than 25 times as features. In all, the data contains 9,797 distinct object strings and 10,151 distinct relation strings that appear at least 25 times.

We judged the precision of each model by manually labeling all of the clusters that each model outputs. Judging recall would require inspecting not just the clusters that the system outputs, but the entire data set, to find all of the true clusters. Because of the size of the data set, we instead estimated recall over a smaller subset of the data. We took the top 200 most frequent object strings and top 200 most frequent relation strings in the data. For each one of these high-frequency strings, we manually searched through all strings with frequency over 25 that shared at least one property, as well as all strings that contained one of the keywords in the high-frequency strings or obvious variations of them. We manually clustered the resulting matches. The top 200 object strings formed 51 clusters of size greater than one, with an average cluster size of 2.9. For relations, the top 200 strings and their matches formed 110 clusters with size greater than one, with an average cluster size of 4.9. We measured the recall of our models by comparing the set of all clusters containing at least one of the high-frequency words against these gold standard clusters.

For our precision and recall measures, we only compare clusters of size two or more, in order to focus on the interesting cases. Using the term *hypothesis cluster* for clusters created by one of the models, we define the precision of a model to be the number of elements in all hypothesis clusters which are correct divided by the total number of elements in hypothesis clusters. An element s is marked correct if a plurality of the elements in s's cluster refer to the same entity as s; we break ties arbitrarily, as they do not affect results. We define recall as the sum over gold standard clusters of the most number of elements found in a single hypothesis cluster, divided by the total number of elements in gold standard clusters.

For the ESP and SSM models in our experiment, we prevented mutual recursion by clustering relations and objects separately. Only the full RESOLVER system uses mutual recursion. For the CSM model, we create for each distinct string a row vector, with each column representing a property. If that property applies to the string, we set the value of that column to the inverse frequency of the property and zero otherwise. CSM finds the cosine of the angle between the vectors for each pair of strings, and merges the best pairs that score above threshold.

Each model requires a threshold parameter to determine which scores are suitable for merging. For these experiments we arbitrarily chose a threshold of 3 for the ESP model (that is, the data needs to be 3 times more likely given the merged cluster than the unmerged clusters in order to perform the merge) and chose thresholds for the other models by hand so that the difference between them and ESP would be roughly even between precision and recall, although for relations it was harder to improve the recall. It is an important item for future work to be able to estimate these thresholds and perhaps other parameters of our models from unlabeled data, but the chosen parameters worked well enough for the experiments. Table 1 shows the precision and recall of our models.

6.1 Discussion

ESP significantly outperforms CSM on both object and relation clustering. CSM had particular trouble with lower-frequency strings, judging far too many of them to be co-referential on too little evidence. If the threshold for clustering using CSM is increased, however, the recall begins to approach zero.

ESP and CSM make predictions based on a very noisy signal. "Canada," for example, shares more properties with "United States" in our data than "U.S." does, even though "Canada" appears less often than "U.S." The results show that both models perform below the SSM model on its own for object merging, and both perform slightly better than SSM on relations because of SSM's poor recall.

We found a significant improvement in both pre-

[6]The data is made available at http://www.cs.washington.edu/homes/ayates/.

	Objects			Relations		
Model	Prec.	Rec.	F1	Prec.	Rec.	F1
CSM	0.51	0.36	0.42	0.62	0.29	0.40
ESP	**0.56**	0.41	0.47	**0.79**	0.33	0.47
SSM	**0.62**	0.53	0.57	**0.85**	0.25	0.39
RESOLVER	**0.71**	0.66	0.68	**0.90**	**0.35**	0.50

Table 1: **Comparison of the cosine similarity metric (CSM), RESOLVER components (SSM and ESP), and the RESOLVER system.** Bold indicates the score is significantly different from the score in the row above at $p < 0.05$ using the chi-squared test with one degree of freedom. Using the same test, RESOLVER is also significantly different from ESP and CSM in recall on objects, and from CSM and SSM in recall on relations. RESOLVER's F1 on objects is a 19% increase over SSM's F1. RESOLVER's F1 on relations is a 28% increase over SSM's F1.

cision and recall when using a combined model over using SSM alone. RESOLVER's F1 is 19% higher than SSM's on objects, and 28% higher on relations.

In a separate experiment we found that mutual recursion provides mixed results. A combination of SSM and ESP without mutual recursion had a precision of 0.76 and recall of 0.59 on objects, and a precision of 0.91 and recall of 0.35 on relations. Mutual recursion increased recall and decreased precision for both objects and relations. None of the differences were statistically significant, however.

There is clearly room for improvement on the SR task. Except for the problem of confusing similar and identical pairs (see section 5), error analysis shows that most of RESOLVER's mistakes are because of two kinds of errors:

1. *Extraction errors*. For example, "US News" gets extracted separately from "World Report", and then RESOLVER clusters them together because they share almost all of the same properties.

2. *Multiple word senses*. For example, there are two President Bushes; also, there are many terms like "President" and "Army" that can refer to many different entities.

6.2 Experiments with Extensions

The extensions to RESOLVER attempt to address the confusion between similar and identical pairs. Experiments with the extensions, using the same datasets and metrics as above, demonstrate that the Function Filter (FF) and the Coordination-Phrase Filter (CPF) boost RESOLVER's performance.

FF requires as input the set of functional and one-to-one relations in the data. Table 2 contains a sam-

is capital of	is capital city of
named after	was named after
headquartered in	is headquartered in

Table 2: **A sample of the set of functions used by the Function Filter.**

Model	Prec.	Rec.	F1
RESOLVER	0.71	0.66	0.68
RESOLVER+FF	0.74	0.66	0.70
RESOLVER+CPF	**0.78**	0.68	0.73
RESOLVER+FF+CPF	**0.78**	0.68	0.73

Table 3: **Comparison of object merging results for the RESOLVER system, RESOLVER plus Function Filtering** (RESOLVER+FF)**, RESOLVER plus Coordination-Phrase Filtering** (RESOLVER+CPF)**, and RESOLVER plus both types of filtering** (RESOLVER+FF+CPF)**.** Bold indicates the score is significantly different from RESOLVER's score at $p < 0.05$ using the chi-squared test with one degree of freedom. RESOLVER+CPF's F1 on objects is a 28% increase over SSM's F1, and a 7% increase over RESOLVER's F1.

pling of the manually-selected functions used in our experiment. Automatically discovering such functions from extractions has been addressed in Ana-Maria Popescu's dissertation (Popescu, 2007), and we did not attempt to duplicate this effort in RESOLVER.

Table 3 contains the results of our experiments. With coordination-phrase filtering, RESOLVER's F1 is 28% higher than SSM's on objects, and 6% higher than RESOLVER's F1 without filtering. While function filtering is a promising idea, FF provides a smaller benefit than CPF on this dataset, and the merges that it prevents are, with a few exceptions, a subset of the merges prevented by CPF. This is in part due to the limited number of functions available in the data. In addition to outperforming FF on this dataset, CPF has the added advantage that it does not require additional input, like a set of functions.

7 Conclusion and Future Work

We have shown that the unsupervised and scalable RESOLVER system is able to find clusters of co-referential object names in extracted relations with a precision of 78% and a recall of 68% with the aid of coordination-phrase filtering, and can find clusters of co-referential relation names with precision of 90% and recall of 35%. We have demonstrated significant improvements over using simple similarity metrics for this task by employing a novel probabilistic model of coreference.

In future work, we plan to use RESOLVER on a much larger data set of over a hundred million assertions, further testing its scalability and its ability to improve in accuracy given additional data. We also plan to add techniques for handling multiple word senses. Finally, to make the probabilistic model more accurate and easier to use, we plan to investigate methods for automatically estimating its parameters from unlabeled data.

Acknowledgements

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-05-1-0185 as well as gifts from Google, and carried out at the University of Washington's Turing Center. We thank Doug Downey, Michele Banko, Stef Schoenmackers, Dan Weld, Fei Wu, and the anonymous reviewers for their helpful comments on previous drafts.

References

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

I. Bhattacharya and L. Getoor. 2005. Relational Clustering for Multi-type Entity Resolution. In *11th ACM SIGKDD Workshop on Multi Relational Data Mining*.

I. Bhattacharya and L. Getoor. 2006. Query-time entity resolution. In *KDD*.

W.W. Cohen, P. Ravikumar, and S.E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *IIWeb*.

I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science*, 3944:177–190.

X. Dong, A.Y. Halevy, and J. Madhavan. 2005. Reference reconciliation in complex information spaces. In *SIGMOD*.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *IJCAI*.

M. A. Hernandez and S. J. Stolfo. 1995. The merge/purge problem for large databases. In *SIGMOD*.

D. Hindle. 1990. Noun classification from predicage-argument structures. In *ACL*.

A. Kehler. 1997. Probabilistic coreference in information extraction. In *EMNLP*.

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

D. Lin and P. Pantel. 2001. DIRT – Discovery of Inference Rules from Text. In *KDD*.

A. McCallum and B. Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*.

A. McCallum, K. Nigam, and L. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*.

Ana-Maria Popescu. 2007. *Information Extraction from Unstructured Web Text*. University of Washington.

P. Ravikumar and W. W. Cohen. 2004. A hierarchical graphical model for record linkage. In *UAI*.

G. Salton and M.J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

P. Singla and P. Domingos. 2006. Entity Resolution with Markov Logic. In *ICDM*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *COLING/ACL*.

W.E. Winkler. 1999. The state of record linkage and current research problems. Technical report, U.S. Bureau of the Census, Washington, D.C.

The Domain Restriction Hypothesis: Relating Term Similarity and Semantic Consistency

Alfio Massimiliano Gliozzo
ITC-irst
Trento, Italy
gliozzo@itc.it

Marco Pennacchiotti
University of Rome Tor Vergata
Rome, Italy
pennacchiotti@info.uniroma2.it

Patrick Pantel
USC, Information Sciences Institute
Marina del Rey, CA
pantel@isi.edu

Abstract

In this paper, we empirically demonstrate what we call the domain restriction hypothesis, claiming that semantically related terms extracted from a corpus tend to be semantically coherent. We apply this hypothesis to define a post-processing module for the output of *Espresso*, a state of the art relation extraction system, showing that irrelevant and erroneous relations can be filtered out by our module, increasing the precision of the final output. Results are confirmed by both quantitative and qualitative analyses, showing that very high precision can be reached.

1 Introduction

Relation extraction is a fundamental step in many natural language processing applications such as learning ontologies from texts (Buitelaar et al., 2005) and Question Answering (Pasca and Harabagiu, 2001).

The most common approach for acquiring concepts, instances and relations is to harvest semantic knowledge from texts. These techniques have been largely explored and today they achieve reasonable accuracy. Harvested lexical resources, such as concept lists (Pantel and Lin, 2002), facts (Etzioni et al., 2002) and semantic relations (Pantel and Pennacchiotti, 2006) could be then successfully used in different frameworks and applications.

The state of the art technology for relation extraction primarily relies on pattern-based approaches (Snow et al., 2006). These techniques are based on the recognition of the typical patterns that express a particular relation in text (e.g. *"X such as Y"* usually expresses an *is-a* relation). Yet, text-based algorithms for relation extraction, in particular pattern-based algorithms, still suffer from a number of limitations due to complexities of natural language, some of which we describe below.

Irrelevant relations. These are valid relations that are not of interest in the domain at hand. For example, in a political domain, *"Condoleezza Rice is a football fan"* is not as relevant as *"Condoleezza Rice is the Secretary of State of the United States"*. Irrelevant relations are ubiquitous, and affect ontology reliability, if used to populate it, as the relation drives the wrong type of ontological knowledge.

Erroneous or false relations. These are particularly harmful, since they directly affect algorithm precision. A pattern-based relation extraction algorithm is particularly likely to extract erroneous relations if it uses *generic patterns*, which are defined in (Pantel and Pennacchiotti, 2006) as broad coverage, noisy patterns with high recall and low precision (e.g. *"X of Y"* for *part-of* relation). Harvesting algorithms either ignore generic patterns (Hearst, 1992) (affecting system recall) or use manually supervised filtering approaches (Girju et al., 2006) or use completely unsupervised Web-filtering methods (Pantel and Pennacchiotti, 2006). Yet, these methods still do not sufficiently mitigate the problem of erroneous relations.

Background knowledge. Another aspect that makes relation harvesting difficult is related to the

nature of semantic relations: relations among entities are mostly paradigmatic (de Saussure, 1922), and are usually established *in absentia* (i.e., they are not made explicit in text). According to Eco's position (Eco, 1979), the background knowledge (e.g. *"persons are humans"*) is often assumed by the writer, and thus is not explicitly mentioned in text. In some cases, such widely-known relations can be captured by distributional similarity techniques but not by pattern-based approaches.
Metaphorical language. Even when paradigmatic relations are explicitly expressed in texts, it can be very difficult to distinguish between facts and metaphoric usage (e.g. the expression *"My mind is a pearl"* occurs 17 times on the Web, but it is clear that *mind* is not a *pearl*, at least from an ontological perspective).

The considerations above outline some of the difficulties of taking a purely lexico-syntactic approach to relation extraction. Pragmatic issues (*background knowledge* and *metaphorical language*) and ontological issues (*irrelevant relation*) can not be solved at the syntactic level. Also, *erroneous relations* can always arise. These considerations lead us to the intuition that extraction can benefit from imposing some additional constraints.

In this paper, we integrate Espresso with a lexical distribution technique modeling semantic coherence through *semantic domains* (Magnini et al., 2002). These are defined as common discourse topics which demonstrate lexical coherence, such as ECONOMICS or POLITICS. We explore whether semantic domains can provide the needed additional constraints to mitigate the acceptance of erroneous relations. At the *lexical level*, semantic domains identify clusters of (domain) paradigmatically related terms. We believe that the main advantage of adopting semantic domains in relation extraction is that relations are established mainly among terms in the same Domain, while concepts belonging to different fields are mostly unrelated (Gliozzo, 2005), as described in Section 2. For example, in a chemistry domain, an *is-a* will tend to relate only terms of that domain (e.g., *nitrogen is-a element*), while out-of-domain relations are likely to be erroneous *e.g., driver is-a element*.
By integrating pattern-based and distributional approaches we aim to capture the two characteristic properties of semantic relations:

- *Syntagmatic properties*: if two terms X and Y are in a given relation, they tend to co-occur in texts, and are mostly connected by specific lexical-syntactic patterns (e.g., the patter "X is a Y" connects terms in *is-a* relations). This aspect is captured using a pattern-based approach.

- *Domain properties*: if a semantic relation among two terms X and Y holds, both X and Y should belong to the same semantic domain (i.e. they are *semantically coherent*), where semantic domains are sets of terms characterized by very similar distributional properties in a (possibly domain specific) corpus.

In Section 2, we develop the concept of semantic domain and an automatic acquisition procedure based on Latent Semantic Analysis (LSA) and we provide empirical evidence of the connection between relation extraction and domain modelling. Section 3 describes the Espresso system. Section 4 concerns our integration of semantic domains and Espresso. In Section 5, we evaluate the impact of our LSA domain restriction module on improving a state of the art relation extraction system. In Section 6 we draw some interesting research directions opened by our work.

2 Semantic Domains

Semantic domains are common areas of human discussion, which demonstrate lexical coherence, such as ECONOMICS, POLITICS, LAW, SCIENCE, (Magnini et al., 2002). At the *lexical level*, semantic domains identify clusters of (domain) related lexical-concepts, i.e. sets of highly paradigmatically related words also known as Semantic Fields.

In the literature, semantic domains have been inferred from corpora by adopting term clustering methodologies (Gliozzo, 2005), and have been used for several NLP tasks, such as Text Categorization and Ontology Learning (Gliozzo, 2006).

Semantic domains can be described by Domain Models (DMs) (Gliozzo, 2005). A DM is a computational model for semantic domains, that represents domain information at the term level, by defining a set of term clusters. Each cluster represents a Semantic Domain, i.e. a set of terms that often cooccur in texts having similar topics. A DM is represented by a $k \times k'$ rectangular matrix $\mathbf{D}$, containing the domain relevance for each term with respect to each domain, as illustrated in Table 1.

	MEDICINE	COMPUTER_SCIENCE
HIV	1	0
AIDS	1	0
virus	0.5	0.5
laptop	0	1

Table 1: Example of a Domain Model

DMs can be acquired from texts in a completely unsupervised way by exploiting a lexical coherence assumption. To this end, term clustering algorithms can be used with each cluster representing a Semantic Domain. The degree of association among terms and clusters, estimated by the learning algorithm, provides a domain relevance function. For our experiments we adopted a clustering strategy based on LSA (Deerwester et al., 1990), following the methodology described in (Gliozzo, 2005). The input of the LSA process is a term-by-document matrix $\mathbf{T}$ reporting the term frequencies in the whole corpus for each term. The matrix is decomposed by means of a Singular Value Decomposition (SVD), identifying the principal components of $\mathbf{T}$. This operation is done off-line, and can be efficiently performed on large corpora. SVD decomposes $\mathbf{T}$ into three matrixes $\mathbf{T} \simeq \mathbf{V}\mathbf{\Sigma_{k'}}\mathbf{U}^T$ where $\mathbf{\Sigma_{k'}}$ is the diagonal $k \times k$ matrix containing the highest $k' \ll k$ eigenvalues of $\mathbf{T}$ on the diagonal, and all the remaining elements are 0. The parameter k' is the dimensionality of the domain and can be fixed in advance[1]. Under this setting we define the domain matrix $\mathbf{D_{LSA}}$[2] as

$$\mathbf{D_{LSA}} = \mathbf{I^N V}\sqrt{\mathbf{\Sigma_{k'}}} \tag{1}$$

where $\mathbf{I^N}$ is a diagonal matrix such that $\mathbf{i^N_{i,i}} = \frac{1}{\sqrt{\langle \vec{w'_i}, \vec{w'_i} \rangle}}$ and $\vec{w'_i}$ is the i^{th} row of the matrix $\mathbf{V}\sqrt{\mathbf{\Sigma_{k'}}}$.

[1]It is not clear how to choose the right dimensionality. In our experiments we used 100 dimensions.

[2]Details of this operation can be found in (Gliozzo, 2005).

Once a DM has been defined by the matrix $\mathbf{D}$, the Domain Space is a k' dimensional space, in which both texts and terms are associated to Domain Vectors (DVs), i.e. vectors representing their domain relevancies with respect to each domain. The DV $\vec{t'_i}$ for the term $t_i \in \mathcal{V}$ is the i^{th} row of $\mathbf{D}$, where $\mathcal{V} = \{t_1, t_2, \ldots, t_k\}$ is the vocabulary of the corpus. The domain similarity $\phi_d(t_i, t_j)$ among terms is then estimated by the cosine among their corresponding DVs in the Domain Space, defined as follows:

$$\phi_d(t_i, t_j) = \frac{\langle \vec{t_i}, \vec{t_j} \rangle}{\sqrt{\langle \vec{t_i}, \vec{t_i} \rangle \langle \vec{t_j}, \vec{t_j} \rangle}} \tag{2}$$

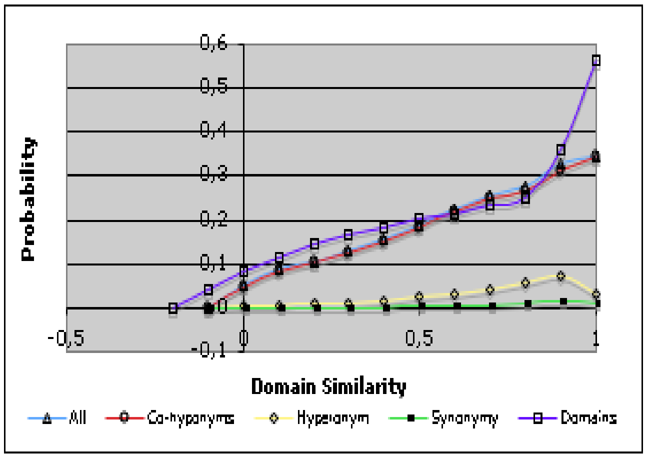


Figure 1: Probability of finding paradigmatic relations

The main advantage of adopting semantic domains for relation extraction is that they allow us to impose a domain restriction on the set of candidate pairs of related terms. In fact, semantic relations can be established mainly among terms in the same Semantic Domain, while concepts belonging to different fields are mostly unrelated.

To show the validity of the domain restriction we conducted a preliminary experiment, contrasting the probability for two words to be related in WordNet (Magnini and Cavaglià, 2000) with their domain similarity, measured in the Domain Space induced from the British National Corpus. In particular, for each couple of words, we estimated the domain similarity, and we collected word pairs in sets characterized by different ranges of similarity (e.g. all the pairs between 0.8 and 0.9). Then we estimated the probability of each couple of words in different sets to be linked by a semantic relation in WordNet, such as synonymy, hyperonymy, co-hyponymy and domain in WordNet Domains (Magnini et al., 2002). Results in Figure 1 show a monotonic crescent relation between these two quantities. In particular the probability for two words to be related tends to 0 when their similarity is negative (i.e., they are not domain related), supporting the basic hypothesis of this work. In Section 4 we will show that this property can be used to improve the overall performances of the relation extraction algorithm.

3 The pattern-based Espresso system

Espresso (Pantel and Pennacchiotti, 2006) is a corpus-based general purpose, broad, and accurate relation extraction algorithm requiring minimal supervision, whose core is based on the framework adopted in (Hearst, 1992). Espresso introduces two main innovations that guarantee high performance: (i) a principled measure for estimating the reliability of relational patterns and instances; (ii) an algorithm for exploiting *generic patterns*. Generic patterns are broad coverage noisy patterns (high recall and low precision), e.g. *"X of Y"* for the *part-of* relation. As underlined in the introduction, previous algorithms either required significant manual work to make use of generic patterns, or simply ignore them. Espresso exploits an unsupervised Web-based filtering method to detect generic patterns and to distinguish their correct and incorrect instances.

Given a specific relation (e.g. *is-a*) and a POS-tagged corpus, Espresso takes as input few seed instances (e.g. *nitrogen is-a element*) or seed surface patterns (e.g. *X/NN such/JJ as/IN Y/NN*). It then incrementally learns new patterns and instances by iterating on the following three phases, until a specific stop condition is met (i.e., new patterns are below a pre-defined threshold of reliability).

Pattern Induction. Given an input set of seed instances I, Espresso infers new patterns connecting as many instances as possible in the given corpus. To do so, Espresso uses a slight modification of the state of the art algorithm described in (Ravichandran and Hovy, 2002). For each instance in input, the sentences containing it are first retrieved and then generalized, by replacing term expressions with a terminological label using regular expressions on the POS-tags. This generalization allows to ease the problem of data sparseness in small corpora. Unfortunately, as patterns become more generic, they are more prone to low precision.

Pattern Ranking and Selection. Espresso ranks all extracted patterns using a *reliability measure* r_π and discards all but the top-k P patterns, where k is set to the number of patterns from the previous iteration plus one. r_π captures the intuition that a reliable pattern is one that is both highly precise and one that extracts many instances. r_π is formally defined as the average strength of association between a pattern p and each input instance i in I, weighted by the reliability r_ι of the instance i (described later):

$$r_\pi(p) = \frac{\sum_{i \in I} \left(\frac{pmi(i,p)}{max_{pmi}} * r_\iota(i) \right)}{|I|}$$

where $pmi(i,p)$ is the pointwise mutual information (pmi) between i and p (estimated with Maximum Likelihood Estimation), and max_{pmi} is the maximum pmi between all patterns and all instances.

Instance Extraction, Ranking, Selection. Espresso extracts from the corpus the set of instances I matching the patterns in P. In this phase generic patterns are detected, and their instances are filtered, using a technique described in detail in (Pantel and Pennacchiotti, 2006). Instances are then ranked using a reliability measure r_ι, similar to that adopted for patterns. A reliable instance should be highly associated with as many reliable patterns as possible:

$$r_\iota(i) = \frac{\sum_{p \in P} \left(\frac{pmi(i,p)}{max_{pmi}} * r_\pi(i) \right)}{|P|}$$

Finally, the best scoring instances are selected for the following iteration. If the number of extracted instances is too low (as often happens in small corpora) Espresso enters an expansion phase, in which instances are expanded by using web based and syntactic techniques.

The output Espresso is a list of instances $i = (X, Y) \in I$, ranked according to $r_\iota(i)$. This score accounts for the *syntagmatic* similarity between X and Y, i.e., how strong is the co-occurrence of X and Y in texts with a given pattern p.

A key role in the Espresso algorithm is played by the reliability measures. The accuracy of the whole extraction process is in fact highly sensitive to the ranking of patterns and instances because, at each iteration, only the best scoring entities are retained. For instance, if an erroneous instance is selected after the first iteration, it could in theory affect the following pattern extraction phase and cause drift in consequent iterations. This issue is critical for generic patterns (where precision is still a problem, even with Web-based filtering), and could sometimes also affect non-generic patterns.

It would be then useful to integrate Espresso with a technique able to retain only very precise instances, without compromising recall. As syntagmatic strategies are already in place, another strategy is needed. In the next Section, we show how this can be achieved using instance domain information.

4 Integrating syntagmatic and domain information

The strategy of integrating syntagmatic and domain information has demonstrated to be fruitful in many NLP tasks, such as Word Sense Disambiguation and open domain Ontology Learning (Gliozzo, 2006). According to the structural view (de Saussure, 1922), both aspects contribute to determine the linguistic value (i.e. the meaning) of words: the meaning of lexical constituents is determined by a complex network of semantic relations among words. This suggests that relation extraction can benefit from accounting for both syntagmatic and domain aspects at the same time.

To demonstrate the validity of this claim we can explore many different integration schemata. For example we can restrict the search space (i.e. the set of candidate instances) to the set of all those terms belonging to the same domain. Another possibility is to exploit a similarity metric for domain relatedness to re-rank the output instances I of Espresso, hoping that the top ranked ones will mostly be those which are correct. One advantage of this latter methodology is that it can be applied to the output of any relation extraction system without any modification to the system itself. In addition, this methodology can be evaluated by adopting standard Information Retrieval (IR) measures, such as mean average precision (see Section 5). Because of these advantages, we decided to adopt the re-ranking procedure.

The procedure is defined as follows: each instance extracted by Espresso is assigned a *Domain Similarity* score $\phi_d(X, Y)$ estimated in the domain space according to Equation 2; a higher score is then assigned to the instances that tend to co-occur in the same documents in the corpus. For example, the candidate instances *ethanol is-a nonaromatic_alcohol* has a higher score than *ethanol is-a something*, as *ethanol* and *alcohol* are both from the chemistry domain, while *something* is a generic term and is thus not associated to any domain.

Instances are then re-ranked according to $\phi_d(X, Y)$, which is used as the new index of reliability instead of the original reliability scores of Espresso. In Subsection 5.2 we will show that the re-ranking technique improves the original reliability scores of Espresso.

5 Evaluation

In this Section we evaluate the benefits of applying the domain information to relation extraction (ESP-LSA), by measuring the improvements of Espresso due to domain based re-ranking.

5.1 Experimental Settings

As a baseline system, we used the ESP- implementation of Espresso described in (Pantel and Pennacchiotti, 2006). ESP- is a fully functioning Espresso system, without the generic pattern filtering module (ESP+). We decided to use ESP- for two main reasons. First, the manual evaluation process would have been too time consuming, as ESP+ extracts thousands of relations. Also, the small scale experiment for EXP- allows us to better analyse and compare the results.

To perform the re-ranking operation, we acquired a Domain Model from the input corpus itself. To this aim we performed a SVD of the term by document matrix T describing the input corpus, indexing all the candidate terms recognized by Espresso.

As an evaluation benchmark, we adopted the same instance sets extracted by ESP- in the experiment described in (Pantel and Pennacchiotti, 2006). We used an input corpus of 313,590 words, a college chemistry textbook (Brown et al. 2003), pre-processed using the Alembic Workbench POS-tagger (Day et al. 1997). We considered the following relations: **is-a**, **part-of**, **reaction** (a relation of chemical reaction among chemical entities) and **production** (a process or chemical element/object producing a result). ESP- extracted 200 *is-a*, 111 *part-of*, 40 *reaction* and 196 *production* instances.

5.2 Quantitative Analysis

The experimental evaluation compared the accuracy of the ranked set of instances extracted by ESP- with the re-ranking produced on these instances by ESP-LSA. By analogy to IR, we are interested in extracting positive instances (i.e. semantically related words). Accordingly, we utilize the standard definitions of precision and recall typically used in IR . Table 2 reports the Mean Average Precision obtained by both ESP- and ESP-LSA on the extracted relations, showing the substantial improvements on all the relations due to domain based re-ranking.

	ESP-	ESP-LSA	
is-a	0.54	**0.75**	(+0.21)
part-of	0.65	**0.82**	(+0.17)
react	0.75	**0.82**	(+0.07)
produce	0.55	**0.62**	(+0.07)

Table 2: Mean Average Precision reported by ESP- and ESP-LSA

Figures 2, 3, 4 and 5 report the precision/recall curves obtained for each relation, estimated by measuring the precision / recall at each point of the ranked list. Results show that precision is very high especially for the top ranked relations extracted by ESP-LSA. Precision reaches the upper bound for the top ranked part of the **part-of** relation, while it is close to 0.9 for the *is-a* relation. In all cases, the precision reported by the ESP-LSA system surpass those of the ESP- system at all recall points.

5.3 Qualitative Analysis

Table 3 shows the best scoring instances for ESP- and ESP-LSA on the evaluated relations. Results

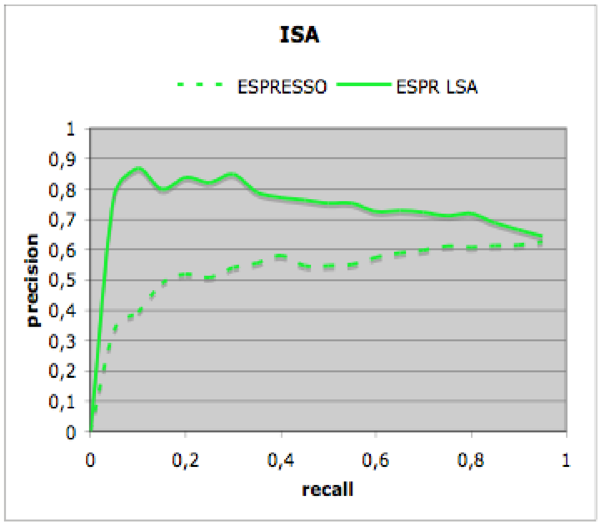


Figure 2: Syntagmatic vs. Domain ranking for the **is-a** relation

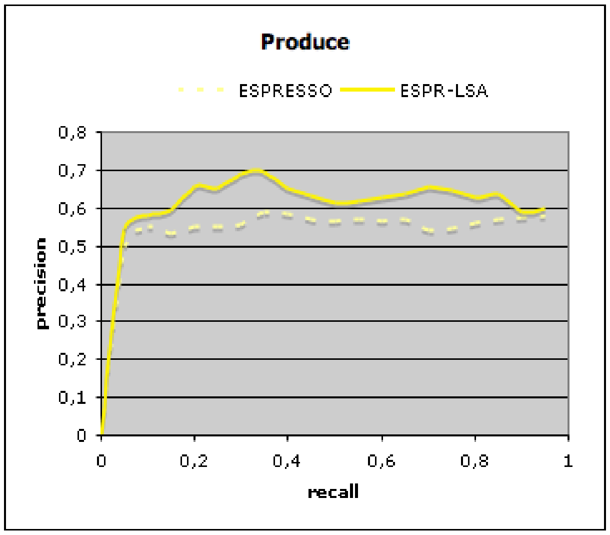


Figure 3: Syntagmatic vs. Domain ranking for the **produce** relation

show that ESP-LSA tends to assign a much lower score to erroneous instances, as compared to the original Espresso reliability ranking. For example for the *part-of* relation, the ESP- ranks the erroneous instance *geometry part-of ion* in 23th position, while ESP-LSA re-ranks it in 92nd. In this case, a lower score is assigned because *geometry* is not particularly tied to the domain of chemistry. Also, ESP-LSA tends to penalize instances derived from parsing/tokenization errors:

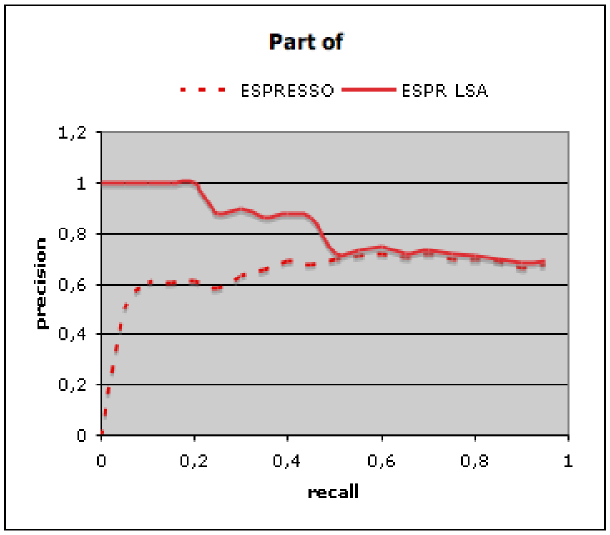


Figure 4: Syntagmatic vs. Domain ranking for the **part-of** relation

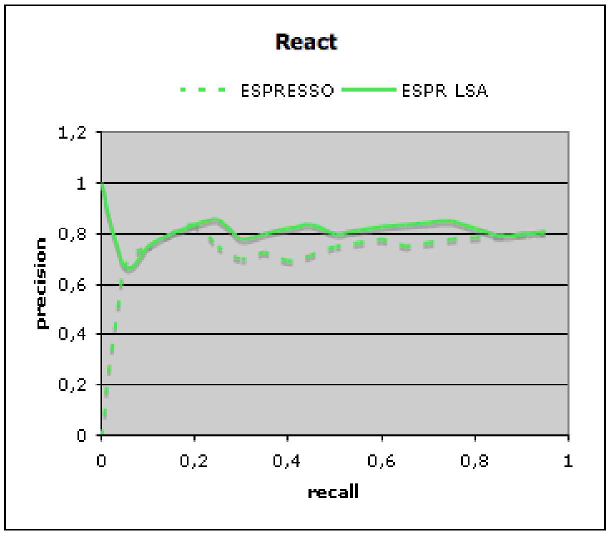


Figure 5: Syntagmatic vs. Domain ranking for the **react** relation

]_binary_hydrogen_compounds_hydrogen react elements is 16th for ESP-, while in the last tenth of the ESP-LSA. In addition, *out-of-domain relations* are successfully interpreted by ESP-LSA. For example, the instance *sentences part-of exceptions* is a possibly correct relation, but unrelated to the domain, as an *exception* in chemistry has nothing to do with *sentences*. This instance lies at the bottom of the ESP-LSA ranking, while is in the middle of ESP- list. Also, low ranked and correct relations extracted by ESP- emerge with ESP-LSA. For example, *magnesium_metal react elemental_oxygen* lies at the end of ESP- rank, as there are not enough syntagmatic evidence (co-occurrences) that let the instance emerge. The domain analysis of ESP-LSA promotes this instance to the 2nd rank position. However, in few cases, the strategy adopted by ESP-LSA tends to promote erroneous instances (e.g. *high voltage produce voltage*). Yet, results show that these are isolated cases.

6 Conclusion and future work

In this paper, we propose the *domain restriction hypothesis*, claiming that semantically related terms extracted from a corpus tend to be semantically coherent. Applying this hypothesis, we presented a new method to improve the precision of pattern-based relation extraction algorithms, where the integration of domain information allows the system to filter out many irrelevant relations, erroneous candidate pairs and metaphorical language relational expressions, while capturing the assumed knowledge required to discover paradigmatic associations among terms. Experimental evidences supports this claim both qualitatively and quantitatively, opening a promising research direction, that we plan to explore much more in depth. In the future, we plan to compare LSA to other term similarity measures, to train the LSA model on large open domain corpora and to apply our technique to both generic and specific corpora in different domains. We want also to increase the level of integration of the LSA technique in the Espresso algorithm, by using LSA as an alternative reliability measure at each iteration. We will also explore the domain restriction property of semantic domains to develop open domain ontology learning systems, as proposed in (Gliozzo, 2006).

The *domain restriction hypothesis* has potential to greatly impact many applications where matching textual expressions is a primary component. It is our hope that by combining existing ranking strategies in applications such as information retrieval, question answering, information extraction and document classification, with knowledge of the coherence of the underlying text, one will see significant improvements in matching accuracy.

Relation	**ESP-**	**ESP - LSA**
X is-a Y	Aluminum ; metal	F ; electronegative_atoms
	nitride_ion ; strong_Br	O ; electronegative_atoms
	heat_flow ; calorimeter	NaCN ; cyanide_salt
	complete_ionic_equation ; spectator	NaCN ; cyanide_salts
X part-of Y	elements ; compound	amino_acid_building_blocks ; tripeptide
	composition ; substance	acid_building_blocks ; tripeptide
	blocks ; tripeptide	powdered_zinc_metal ; battery
	elements ; sodium_chloride	building_blocks ; tripeptide
X react Y	hydrazine ; water	magnesium_metal ; elemental_oxygen
	magnesium_metal ; hydrochloric_acid	nitrogen ; ammonia
	magnesium ; oxygen	sodium_metal ; chloride
	magnesium_metal ; acid	carbon_dioxide ; methane
X produce Y	bromine ; bromide	high_voltage ; voltage
	oxygen ; oxide	reactions ; reactions
	common_fuels ; dioxide	dr_jekyll ; hyde
	kidneys ; stones	yellow_pigments ; green_pigment

Table 3: Top scoring relations extracted by ESP- and ESP-LSA.

Acknowledgments

Thanks to Roberto Basili for his precious comments, suggestions and support. Alfio Gliozzo was supported by the OntoText project, funded by the Autonomous Province of Trento under the FUP-2004, and the FIRB founded project N.RBIN045PXH.

References

P. Buitelaar, P. Cimiano, and B. Magnini. 2005. *Ontology learning from texts: methods, evaluation and applications*. IOS Press.

F. de Saussure. 1922. *Cours de linguistique générale*. Payot, Paris.

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*.

U. Eco. 1979. *Lector in fabula*. Bompiani.

O. Etzioni, M.J. Cafarella, D. Downey, A.-M A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. 2002. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–143.

R. Girju, A. Badulescu, and D. Moldovan. 2006. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of HLT/NAACL-03*, pages 80–87, Edmonton, Canada, July.

A. Gliozzo. 2005. *Semantic Domains in Computational Linguistics*. Ph.D. thesis, University of Trento.

A. Gliozzo. 2006. The god model. In *Proceedings of EACL*.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics. Nantes, France*.

B. Magnini and G. Cavaglià. 2000. Integrating subject field codes into WordNet. In *Proceedings of LREC-2000*, pages 1413–1418, Athens, Greece, June.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

P. Pantel and D. Lin. 2002. Discovering word senses from text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining*, pages 613–619.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL-COLING-06*, pages 113–120, Sydney, Australia.

M. Pasca and S. Harabagiu. 2001. The informative role of wordnet in open-domain question answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*, pages 138–143, Pittsburgh, PA.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL-02*, pages 41–47, Philadelphia, PA.

R. Snow, D. Jurafsky, and A.Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the ACL/COLING-06*, pages 801–808, Sydney, Australia.

Bayesian Inference for PCFGs via Markov chain Monte Carlo

Mark Johnson
Cognitive and Linguistic Sciences
Brown University
Mark_Johnson@brown.edu

Thomas L. Griffiths
Department of Psychology
University of California, Berkeley
Tom_Griffiths@berkeley.edu

Sharon Goldwater
Department of Linguistics
Stanford University
sgwater@stanford.edu

Abstract

This paper presents two Markov chain Monte Carlo (MCMC) algorithms for Bayesian inference of probabilistic context free grammars (PCFGs) from terminal strings, providing an alternative to maximum-likelihood estimation using the Inside-Outside algorithm. We illustrate these methods by estimating a sparse grammar describing the morphology of the Bantu language Sesotho, demonstrating that with suitable priors Bayesian techniques can infer linguistic structure in situations where maximum likelihood methods such as the Inside-Outside algorithm only produce a trivial grammar.

1 Introduction

The standard methods for inferring the parameters of probabilistic models in computational linguistics are based on the principle of maximum-likelihood estimation; for example, the parameters of Probabilistic Context-Free Grammars (PCFGs) are typically estimated from strings of terminals using the Inside-Outside (IO) algorithm, an instance of the Expectation Maximization (EM) procedure (Lari and Young, 1990). However, much recent work in machine learning and statistics has turned away from maximum-likelihood in favor of Bayesian methods, and there is increasing interest in Bayesian methods in computational linguistics as well (Finkel et al., 2006). This paper presents two Markov chain Monte Carlo (MCMC) algorithms for inferring PCFGs and their parses from strings alone. These can be viewed as Bayesian alternatives to the IO algorithm.

The goal of Bayesian inference is to compute a distribution over plausible parameter values. This "posterior" distribution is obtained by combining the likelihood with a "prior" distribution $\mathrm{P}(\theta)$ over parameter values θ. In the case of PCFG inference θ is the vector of rule probabilities, and the prior might assert a preference for a sparse grammar (see below). The posterior probability of each value of θ is given by Bayes' rule:

$$\mathrm{P}(\theta|D) \quad \propto \quad \mathrm{P}(D|\theta)\mathrm{P}(\theta). \tag{1}$$

In principle Equation 1 defines the posterior probability of any value of θ, but computing this may not be tractable analytically or numerically. For this reason a variety of methods have been developed to support approximate Bayesian inference. One of the most popular methods is Markov chain Monte Carlo (MCMC), in which a Markov chain is used to sample from the posterior distribution.

This paper presents two new MCMC algorithms for inferring the posterior distribution over parses and rule probabilities given a corpus of strings. The first algorithm is a component-wise Gibbs sampler which is very similar in spirit to the EM algorithm, drawing parse trees conditioned on the current parameter values and then sampling the parameters conditioned on the current set of parse trees. The second algorithm is a component-wise Hastings sampler that "collapses" the probabilistic model, integrating over the rule probabilities of the PCFG, with the goal of speeding convergence. Both algorithms use an efficient dynamic programming technique to sample parse trees.

Given their usefulness in other disciplines, we believe that Bayesian methods like these are likely to be of general utility in computational linguistics as well. As a simple illustrative example, we use these methods to infer morphological parses for verbs from Sesotho, a southern Bantu language with agglutinating morphology. Our results illustrate that Bayesian inference using a prior that favors sparsity can produce linguistically reasonable analyses in situations in which EM does not.

The rest of this paper is structured as follows. The next section introduces the background for our paper, summarizing the key ideas behind PCFGs, Bayesian inference, and MCMC. Section 3 introduces our first MCMC algorithm, a Gibbs sampler for PCFGs. Section 4 describes an algorithm for sampling trees from the distribution over trees defined by a PCFG. Section 5 shows how to integrate out the rule weight parameters θ in a PCFG, allowing us to sample directly from the posterior distribution over parses for a corpus of strings. Finally, Section 6 illustrates these methods in learning Sesotho morphology.

2 Background

2.1 Probabilistic context-free grammars

Let $G = (T, N, S, R)$ be a Context-Free Grammar in Chomsky normal form with no useless productions, where T is a finite set of *terminal symbols*, N is a finite set of *nonterminal symbols* (disjoint from T), $S \in N$ is a distinguished nonterminal called the *start symbol*, and R is a finite set of *productions* of the form $A \to B\,C$ or $A \to w$, where $A, B, C \in N$ and $w \in T$. In what follows we use β as a variable ranging over $(N \times N) \cup T$.

A *Probabilistic Context-Free Grammar* (G, θ) is a pair consisting of a context-free grammar G and a real-valued vector θ of length $|R|$ indexed by productions, where $\theta_{A\to\beta}$ is the *production probability* associated with the production $A \to \beta \in R$. We require that $\theta_{A\to\beta} \geq 0$ and that for all nonterminals $A \in N$, $\sum_{A\to\beta\in R} \theta_{A\to\beta} = 1$.

A PCFG (G, θ) defines a probability distribution over trees t as follows:

$$\mathrm{P}_G(t|\theta) \;=\; \prod_{r\in R} \theta_r^{f_r(t)}$$

where t is generated by G and $f_r(t)$ is the number of times the production $r = A \to \beta \in R$ is used in the derivation of t. If G does not generate t let $\mathrm{P}_G(t|\theta) = 0$. The *yield* $y(t)$ of a parse tree t is the sequence of terminals labeling its leaves. The probability of a string $w \in T^+$ of terminals is the sum of the probability of all trees with yield w, i.e.:

$$\mathrm{P}_G(w|\theta) \;=\; \sum_{t:y(t)=w} \mathrm{P}_G(t|\theta).$$

2.2 Bayesian inference for PCFGs

Given a corpus of strings $\mathbf{w} = (w_1, \ldots, w_n)$, where each w_i is a string of terminals generated by a known CFG G, we would like to be able to infer the production probabilities θ that best describe that corpus. Taking $\mathbf{w}$ to be our data, we can apply Bayes' rule (Equation 1) to obtain:

$$\begin{aligned}\mathrm{P}(\theta|\mathbf{w}) &\propto \mathrm{P}_G(\mathbf{w}|\theta)\mathrm{P}(\theta), \quad \text{where} \\ \mathrm{P}_G(\mathbf{w}|\theta) &= \prod_{i=1}^{n} \mathrm{P}_G(w_i|\theta).\end{aligned}$$

Using $\mathbf{t}$ to denote a sequence of parse trees for $\mathbf{w}$, we can compute the joint posterior distribution over $\mathbf{t}$ and θ, and then marginalize over $\mathbf{t}$, with $\mathrm{P}(\theta|\mathbf{w}) = \sum_{\mathbf{t}} \mathrm{P}(\mathbf{t}, \theta|\mathbf{w})$. The joint posterior distribution on $\mathbf{t}$ and θ is given by:

$$\begin{aligned}\mathrm{P}(\mathbf{t}, \theta|\mathbf{w}) &\propto \mathrm{P}(\mathbf{w}|\mathbf{t})\mathrm{P}(\mathbf{t}|\theta)\mathrm{P}(\theta) \\ &= \left(\prod_{i=1}^{n} \mathrm{P}(w_i|t_i)\mathrm{P}(t_i|\theta)\right)\mathrm{P}(\theta)\end{aligned}$$

with $\mathrm{P}(w_i|t_i) = 1$ if $y(t_i) = w_i$, and 0 otherwise.

2.3 Dirichlet priors

The first step towards computing the posterior distribution is to define a prior on θ. We take $\mathrm{P}(\theta)$ to be a product of Dirichlet distributions, with one distribution for each non-terminal $A \in N$. The prior is parameterized by a positive real valued vector α indexed by productions R, so each production probability $\theta_{A\to\beta}$ has a corresponding Dirichlet parameter $\alpha_{A\to\beta}$. Let R_A be the set of productions in R

with left-hand side A, and let θ_A and α_A refer to the component subvectors of θ and α respectively indexed by productions in R_A. The Dirichlet prior $\mathrm{P}_D(\theta|\alpha)$ is:

$$\begin{aligned}
\mathrm{P}_D(\theta|\alpha) &= \prod_{A\in N} \mathrm{P}_D(\theta_A|\alpha_A), \quad \text{where} \\
\mathrm{P}_D(\theta_A|\alpha_A) &= \frac{1}{C(\alpha_A)} \prod_{r\in R_A} \theta_r^{\alpha_r - 1} \quad \text{and} \\
C(\alpha_A) &= \frac{\prod_{r\in R_A} \Gamma(\alpha_r)}{\Gamma(\sum_{r\in R_A} \alpha_r)} \qquad (2)
\end{aligned}$$

where Γ is the generalized factorial function and $C(\alpha)$ is a normalization constant that does not depend on θ_A.

Dirichlet priors are useful because they are *conjugate* to the distribution over trees defined by a PCFG. This means that the posterior distribution on θ given a set of parse trees, $\mathrm{P}(\theta|\mathbf{t},\alpha)$, is also a Dirichlet distribution. Applying Bayes' rule,

$$\begin{aligned}
\mathrm{P}_G(\theta|\mathbf{t},\alpha) &\propto \mathrm{P}_G(\mathbf{t}|\theta)\,\mathrm{P}_D(\theta|\alpha) \\
&\propto \left(\prod_{r\in R} \theta_r^{f_r(\mathbf{t})}\right)\left(\prod_{r\in R} \theta_r^{\alpha_r-1}\right) \\
&= \prod_{r\in R} \theta_r^{f_r(\mathbf{t})+\alpha_r-1}
\end{aligned}$$

which is a Dirichlet distribution with parameters $\mathbf{f}(\mathbf{t}) + \alpha$, where $\mathbf{f}(\mathbf{t})$ is the vector of production counts in $\mathbf{t}$ indexed by $r \in R$. We can thus write:

$$\mathrm{P}_G(\theta|\mathbf{t},\alpha) = \mathrm{P}_D(\theta|\mathbf{f}(\mathbf{t}) + \alpha)$$

which makes it clear that the production counts combine directly with the parameters of the prior.

2.4 Markov chain Monte Carlo

Having defined a prior on θ, the posterior distribution over $\mathbf{t}$ and θ is fully determined by a corpus $\mathbf{w}$. Unfortunately, computing the posterior probability of even a single choice of $\mathbf{t}$ and θ is intractable, as evaluating the normalizing constant for this distribution requires summing over all possible parses for the entire corpus and all sets of production probabilities. Nonetheless, it is possible to define algorithms that sample from this distribution using Markov chain Monte Carlo (MCMC).

MCMC algorithms construct a Markov chain whose states $s \in \mathcal{S}$ are the objects we wish to sample. The state space $\mathcal{S}$ is typically astronomically large — in our case, the state space includes all possible parses of the entire training corpus $\mathbf{w}$ — and the transition probabilities $\mathrm{P}(s'|s)$ are specified via a scheme guaranteed to converge to the desired distribution $\pi(s)$ (in our case, the posterior distribution). We "run" the Markov chain (i.e., starting in initial state s_0, sample a state s_1 from $\mathrm{P}(s'|s_0)$, then sample state s_2 from $\mathrm{P}(s'|s_1)$, and so on), with the probability that the Markov chain is in a particular state, $\mathrm{P}(s_i)$, converging to $\pi(s_i)$ as $i \to \infty$.

After the chain has run long enough for it to approach its stationary distribution, the expectation $\mathrm{E}_\pi[f]$ of any function $f(s)$ of the state s will be approximated by the average of that function over the set of sample states produced by the algorithm. For example, in our case, given samples (t_i, θ_i) for $i = 1, \ldots, \ell$ produced by an MCMC algorithm, we can estimate θ as

$$\mathrm{E}_\pi[\theta] \approx \frac{1}{\ell}\sum_{i=1}^{\ell} \theta_i$$

The remainder of this paper presents two MCMC algorithms for PCFGs. Both algorithms proceed by setting the initial state of the Markov chain to a guess for $(\mathbf{t},\theta)$ and then sampling successive states using a particular transition matrix. The key difference betwen the two algorithms is the form of the transition matrix they assume.

3 A Gibbs sampler for $\mathrm{P}(\mathbf{t},\theta|\mathbf{w},\alpha)$

The Gibbs sampler (Geman and Geman, 1984) is one of the simplest MCMC methods, in which transitions between states of the Markov chain result from sampling each component of the state conditioned on the current value of all other variables. In our case, this means alternating between sampling from two distributions:

$$\begin{aligned}
\mathrm{P}(\mathbf{t}|\theta,\mathbf{w},\alpha) &= \prod_{i=1}^{n} \mathrm{P}(t_i|w_i,\theta), \quad \text{and} \\
\mathrm{P}(\theta|\mathbf{t},\mathbf{w},\alpha) &= \mathrm{P}_D(\theta|\mathbf{f}(\mathbf{t})+\alpha) \\
&= \prod_{A\in N} \mathrm{P}_D(\theta_A|\mathbf{f}_A(\mathbf{t})+\alpha_A).
\end{aligned}$$

Thus every two steps we generate a new sample of $\mathbf{t}$ and θ. This alternation between parsing and updating θ is reminiscent of the EM algorithm, with

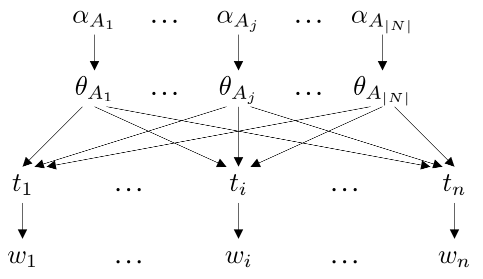

α_{A_1} … α_{A_j} … $\alpha_{A_{|N|}}$
θ_{A_1} … θ_{A_j} … $\theta_{A_{|N|}}$
t_1 … t_i … t_n
w_1 … w_i … w_n

Figure 1: A Bayes net representation of dependencies among the variables in a PCFG.

the Expectation step replaced by sampling $\mathbf{t}$ and the Maximization step replaced by sampling θ.

The dependencies among variables in a PCFG are depicted graphically in Figure 1, which makes clear that the Gibbs sampler is highly parallelizable (just like the EM algorithm). Specifically, the parses t_i are independent given θ and so can be sampled in parallel from the following distribution as described in the next section.

$$\mathrm{P}_G(t_i|w_i,\theta) \quad = \quad \frac{\mathrm{P}_G(t_i|\theta)}{\mathrm{P}_G(w_i|\theta)}$$

We make use of the fact that the posterior is a product of independent Dirichlet distributions in order to sample θ from $\mathrm{P}_D(\theta|\mathbf{t},\alpha)$. The production probabilities θ_A for each nonterminal $A \in N$ are sampled from a Dirchlet distibution with parameters $\alpha'_A = f_A(\mathbf{t}) + \alpha_A$. There are several methods for sampling $\theta = (\theta_1,\ldots,\theta_m)$ from a Dirichlet distribution with parameters $\alpha = (\alpha_1,\ldots,\alpha_m)$, with the simplest being sampling x_j from a Gamma(α_j) distribution for $j = 1,\ldots,m$ and then setting $\theta_j = x_j / \sum_{k=1}^{m} x_k$ (Gentle, 2003).

4 Efficiently sampling from $\mathrm{P}(t|w,\theta)$

This section completes the description of the Gibbs sampler for $(\mathbf{t},\theta)$ by describing a dynamic programming algorithm for sampling trees from the set of parses for a string generated by a PCFG. This algorithm appears fairly widely known: it was described by Goodman (1998) and Finkel et al (2006) and used by Ding et al (2005), and is very similar to other dynamic programming algorithms for CFGs, so we only summarize it here. The algorithm consists of two steps. The first step constructs a standard "inside" table or chart, as used in the Inside-Outside algorithm for PCFGs (Lari and Young, 1990). The second step involves a recursion from larger to smaller strings, sampling from the productions that expand each string and constructing the corresponding tree in a top-down fashion.

In this section we take w to be a string of terminal symbols $w = (w_1,\ldots,w_n)$ where each $w_i \in T$, and define $w_{i,k} = (w_{i+1},\ldots,w_k)$ (i.e., the substring from w_{i+1} up to w_k). Further, let $G_A = (T,N,A,R)$, i.e., a CFG just like G except that the start symbol has been replaced with A, so, $\mathrm{P}_{G_A}(t|\theta)$ is the probability of a tree t whose root node is labeled A and $\mathrm{P}_{G_A}(w|\theta)$ is the sum of the probabilities of all trees whose root nodes are labeled A with yield w.

The Inside algorithm takes as input a PCFG (G,θ) and a string $w = w_{0,n}$ and constructs a table with entries $p_{A,i,k}$ for each $A \in N$ and $0 \leq i < k \leq n$, where $p_{A,i,k} = \mathrm{P}_{G_A}(w_{i,k}|\theta)$, i.e., the probability of A rewriting to $w_{i,k}$. The table entries are recursively defined below, and computed by enumerating all feasible i,k and A in any order such that all smaller values of $k-i$ are enumerated before any larger values.

$$\begin{aligned} p_{A,k-1,k} &= \theta_{A\to w_k} \\ p_{A,i,k} &= \sum_{A\to B\,C\in R} \sum_{i<j<k} \theta_{A\to B\,C}\, p_{B,i,j}\, p_{C,j,k} \end{aligned}$$

for all $A,B,C \in N$ and $0 \leq i < j < k \leq n$. At the end of the Inside algorithm, $\mathrm{P}_G(w|\theta) = p_{S,0,n}$.

The second step of the sampling algorithm uses the function SAMPLE, which returns a sample from $\mathrm{P}_G(t|w,\theta)$ given the PCFG (G,θ) and the inside table $p_{A,i,k}$. SAMPLE takes as arguments a nonterminal $A \in N$ and a pair of string positions $0 \leq i < k \leq n$ and returns a tree drawn from $\mathrm{P}_{G_A}(t|w_{i,k},\theta)$. It functions in a top-down fashion, selecting the production $A \to B\,C$ to expand the A, and then recursively calling itself to expand B and C respectively.

```
function SAMPLE$(A,i,k)$ :
if $k-i=1$ then return TREE$(A,w_k)$
$(j,B,C)$ = MULTI$(A,i,k)$
return TREE$(A$, SAMPLE$(B,i,j)$, SAMPLE$(C,j,k))$
```

In this pseudo-code, TREE is a function that constructs unary or binary tree nodes respectively, and

MULTI is a function that produces samples from a multinomial distribution over the possible "split" positions j and nonterminal children B and C, where:

$$\mathrm{P}(j, B, C) = \frac{\theta_{A \to B\,C}\,\mathrm{P}_{G_B}(w_{i,j}|\theta)\,\mathrm{P}_{G_C}(w_{j,k}|\theta)}{\mathrm{P}_{G_A}(w_{i,k}|\theta)}$$

5 A Hastings sampler for $\mathrm{P}(\mathbf{t}|\mathbf{w}, \alpha)$

The Gibbs sampler described in Section 3 has the disadvantage that each sample of θ requires reparsing the training corpus $\mathbf{w}$. In this section, we describe a component-wise Hastings algorithm for sampling directly from $\mathrm{P}(\mathbf{t}|\mathbf{w}, \alpha)$, marginalizing over the production probabilities θ. Transitions between states are produced by sampling parses t_i from $\mathrm{P}(t_i|w_i, \mathbf{t}_{-i}, \alpha)$ for each string w_i in turn, where $\mathbf{t}_{-i} = (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$ is the current set of parses for $\mathbf{w}_{-i} = (w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n)$. Marginalizing over θ effectively means that the production probabilities are updated after each sentence is parsed, so it is reasonable to expect that this algorithm will converge faster than the Gibbs sampler described earlier. While the sampler does not explicitly provide samples of θ, the results outlined in Sections 2.3 and 3 can be used to sample the posterior distribution over θ for each sample of $\mathbf{t}$ if required.

Let $\mathrm{P}_D(\theta|\alpha)$ be a Dirichlet product prior, and let Δ be the probability simplex for θ. Then by integrating over the posterior Dirichlet distributions we have:

$$\begin{aligned} \mathrm{P}(\mathbf{t}|\alpha) &= \int_\Delta \mathrm{P}_G(\mathbf{t}|\theta)\mathrm{P}_D(\theta|\alpha)d\theta \\ &= \prod_{A \in N} \frac{C(\alpha_A + \mathbf{f}_A(\mathbf{t}))}{C(\alpha_A)} \end{aligned} \qquad (3)$$

where C was defined in Equation 2. Because we are marginalizing over θ, the trees t_i become dependent upon one another. Intuitively, this is because w_i may provide information about θ that influences how some other string w_j should be parsed.

We can use Equation 3 to compute the conditional probability $\mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha)$ as follows:

$$\begin{aligned} \mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha) &= \frac{\mathrm{P}(\mathbf{t}|\alpha)}{\mathrm{P}(\mathbf{t}_{-i}|\alpha)} \\ &= \prod_{A \in N} \frac{C(\alpha_A + \mathbf{f}_A(\mathbf{t}))}{C(\alpha_A + \mathbf{f}_A(\mathbf{t}_{-i}))} \end{aligned}$$

Now, if we could sample from

$$\mathrm{P}(t_i|w_i, \mathbf{t}_{-i}, \alpha) = \frac{\mathrm{P}(w_i|t_i)\mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha)}{\mathrm{P}(w_i|\mathbf{t}_{-i}, \alpha)}$$

we could construct a Gibbs sampler whose states were the parse trees $\mathbf{t}$. Unfortunately, we don't even know if there is an efficient algorithm for calculating $\mathrm{P}(w_i|\mathbf{t}_{-i}, \alpha)$, let alone an efficient sampling algorithm for this distribution.

Fortunately, this difficulty is not fatal. A Hastings sampler for a probability distribution $\pi(s)$ is an MCMC algorithm that makes use of a *proposal distribution* $Q(s'|s)$ from which it draws samples, and uses an acceptance/rejection scheme to define a transition kernel with the desired distribution $\pi(s)$. Specifically, given the current state s, a sample $s' \neq s$ drawn from $Q(s'|s)$ is accepted as the next state with probability

$$A(s, s') = \min\left\{1, \frac{\pi(s')Q(s|s')}{\pi(s)Q(s'|s)}\right\}$$

and with probability $1 - A(s, s')$ the proposal is rejected and the next state is the current state s.

We use a component-wise proposal distribution, generating new proposed values for t_i, where i is chosen at random. Our proposal distribution is the posterior distribution over parse trees generated by the PCFG with grammar G and production probabilities θ', where θ' is chosen based on the current $\mathbf{t}_{-i}$ as described below. Each step of our Hastings sampler is as follows. First, we compute θ' from $\mathbf{t}_{-i}$ as described below. Then we sample t'_i from $\mathrm{P}(t_i|w_i, \theta')$ using the algorithm described in Section 4. Finally, we accept the proposal t'_i given the old parse t_i for w_i with probability:

$$\begin{aligned} A(t_i, t'_i) &= \min\left\{1, \frac{\mathrm{P}(t'_i|w_i, \mathbf{t}_{-i}, \alpha)\mathrm{P}(t_i|w_i, \theta')}{\mathrm{P}(t_i|w_i, \mathbf{t}_{-i}, \alpha)\mathrm{P}(t'_i|w_i, \theta')}\right\} \\ &= \min\left\{1, \frac{\mathrm{P}(t'_i|\mathbf{t}_{-i}, \alpha)\mathrm{P}(t_i|w_i, \theta')}{\mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha)\mathrm{P}(t'_i|w_i, \theta')}\right\} \end{aligned}$$

The key advantage of the Hastings sampler over the Gibbs sampler here is that because the acceptance probability is a ratio of probabilities, the difficult to compute $\mathrm{P}(w_i|\mathbf{t}_{-i},\alpha)$ is a common factor of both the numerator and denominator, and hence is not required. The $P(w_i|t_i)$ term also disappears, being 1 for both the numerator and the denominator since our proposal distribution can only generate trees for which w_i is the yield.

All that remains is to specify the production probabilities θ' of the proposal distribution $\mathrm{P}(t'_i|w_i,\theta')$. While the acceptance rule used in the Hastings algorithm ensures that it produces samples from $\mathrm{P}(t_i|w_i,\mathbf{t}_{-i},\alpha)$ with any proposal grammar θ' in which all productions have nonzero probability, the algorithm is more efficient (i.e., fewer proposals are rejected) if the proposal distribution is close to the distribution to be sampled.

Given the observations above about the correspondence between terms in $\mathrm{P}(t_i|\mathbf{t}_{-i},\alpha)$ and the relative frequency of the corresponding productions in $\mathbf{t}_{-i}$, we set θ' to the expected value $\mathrm{E}[\theta|\mathbf{t}_{-i},\alpha]$ of θ given $\mathbf{t}_{-i}$ and α as follows:

$$\theta'_r = \frac{f_r(\mathbf{t}_{-i})+\alpha_r}{\sum_{r'\in R_A} f_{r'}(\mathbf{t}_{-i})+\alpha_{r'}}$$

6 Inferring sparse grammars

As stated in the introduction, the primary contribution of this paper is introducing MCMC methods for Bayesian inference to computational linguistics. Bayesian inference using MCMC is a technique of generic utility, much like Expectation-Maximization and other general inference techniques, and we believe that it belongs in every computational linguist's toolbox alongside these other techniques.

Inferring a PCFG to describe the syntactic structure of a natural language is an obvious application of grammar inference techniques, and it is well-known that PCFG inference using maximum-likelihood techniques such as the Inside-Outside (IO) algorithm, a dynamic programming Expectation-Maximization (EM) algorithm for PCFGs, performs extremely poorly on such tasks. We have applied the Bayesian MCMC methods described here to such problems and obtain results very similar to those produced using IO. We believe that the primary reason why both IO and the Bayesian methods perform so poorly on this task is that simple PCFGs are not accurate models of English syntactic structure. We know that PCFGs

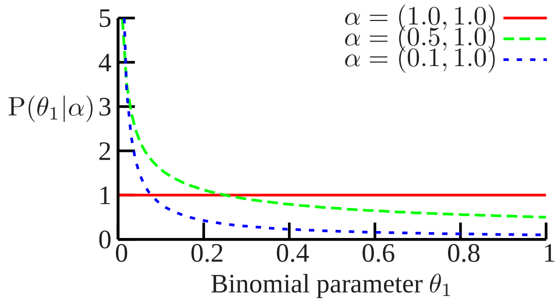


Figure 2: A Dirichlet prior α on a binomial parameter θ_1. As $\alpha_1 \to 0$, $\mathrm{P}(\theta_1|\alpha)$ is increasingly concentrated around 0.

that represent only major phrasal categories ignore a wide variety of lexical and syntactic dependencies in natural language. State-of-the-art systems for unsupervised syntactic structure induction system uses models that are very different to these kinds of PCFGs (Klein and Manning, 2004; Smith and Eisner, 2006).[1]

Our goal in this section is modest: we aim merely to provide an illustrative example of Bayesian inference using MCMC. As Figure 2 shows, when the Dirichlet prior parameter α_r approaches 0 the prior probability $\mathrm{P}_D(\theta_r|\alpha)$ becomes increasingly concentrated around 0. This ability to bias the sampler toward sparse grammars (i.e., grammars in which many productions have probabilities close to 0) is useful when we attempt to identify relevant productions from a much larger set of possible productions via parameter estimation.

The Bantu language Sesotho is a richly agglutinative language, in which verbs consist of a sequence of morphemes, including optional Subject Markers (SM), Tense (T), Object Markers (OM), Mood (M) and derivational affixes as well as the obligatory Verb stem (V), as shown in the following example:

re	*-a*	*-di*	*-bon*	*-a*
SM	T	OM	V	M

"We see them"

[1]It is easy to demonstrate that the poor quality of the PCFG models is the cause of these problems rather than search or other algorithmic issues. If one initializes either the IO or Bayesian estimation procedures with treebank parses and then runs the procedure using the yields alone, the accuracy of the parses uniformly decreases while the (posterior) likelihood uniformly increases with each iteration, demonstrating that improving the (posterior) likelihood of such models does not improve parse accuracy.

We used an implementation of the Hastings sampler described in Section 5 to infer morphological parses $\mathbf{t}$ for a corpus $\mathbf{w}$ of 2,283 unsegmented Sesotho verb types extracted from the Sesotho corpus available from CHILDES (MacWhinney and Snow, 1985; Demuth, 1992). We chose this corpus because the words have been morphologically segmented manually, making it possible for us to evaluate the morphological parses produced by our system. We constructed a CFG G containing the following productions

$$\begin{array}{lcl} \mathsf{Word} & \rightarrow & \mathsf{V} \\ \mathsf{Word} & \rightarrow & \mathsf{V\ M} \\ \mathsf{Word} & \rightarrow & \mathsf{SM\ V\ M} \\ \mathsf{Word} & \rightarrow & \mathsf{SM\ T\ V\ M} \\ \mathsf{Word} & \rightarrow & \mathsf{SM\ T\ OM\ V\ M} \end{array}$$

together with productions expanding the preterminals $\mathsf{SM}, \mathsf{T}, \mathsf{OM}, \mathsf{V}$ and M to each of the 16,350 distinct substrings occuring anywhere in the corpus, producting a grammar with 81,755 productions in all. In effect, G encodes the basic morphological structure of the Sesotho verb (ignoring factors such as derivation morphology and irregular forms), but provides no information about the phonological identity of the morphemes.

Note that G actually generates a *finite* language. However, G parameterizes the probability distribution over the strings it generates in a manner that would be difficult to succintly characterize except in terms of the productions given above. Moreover, with approximately 20 times more productions than training strings, each string is highly ambiguous and estimation is highly underconstrained, so it provides an excellent test-bed for sparse priors.

We estimated the morphological parses $\mathbf{t}$ in two ways. First, we ran the IO algorithm initialized with a uniform initial estimate θ_0 for θ to produce an estimate of the MLE $\hat{\theta}$, and then computed the Viterbi parses $\hat{\mathbf{t}}$ of the training corpus $\mathbf{w}$ with respect to the PCFG $(G, \hat{\theta})$. Second, we ran the Hastings sampler initialized with trees sampled from (G, θ_0) with several different values for the parameters of the prior. We experimented with a number of techniques for speeding convergence of both the IO and Hastings algorithms, and two of these were particularly effective on this problem. Annealing, i.e., using $\mathrm{P}(\mathbf{t}|\mathbf{w})^{1/\tau}$ in place of $\mathrm{P}(\mathbf{t}|\mathbf{w})$ where τ is a "temperature" parameter starting around 5 and slowly adjusted toward 1, sped the convergence of both algorithms. We ran both algorithms for several thousand iterations over the corpus, and both seemed to converge fairly quickly once τ was set to 1. "Jittering" the initial estimate of θ used in the IO algorithm also sped its convergence.

The IO algorithm converges to a solution where $\theta_{\mathsf{Word} \rightarrow \mathsf{V}} = 1$, and every string $w \in \mathbf{w}$ is analysed as a single morpheme V. (In fact, in this grammar $\mathrm{P}(w_i|\theta)$ is the empirical probability of w_i, and it is easy to prove that this θ is the MLE).

The samples $\mathbf{t}$ produced by the Hastings algorithm depend on the parameters of the Dirichlet prior. We set α_r to a single value α for all productions r. We found that for $\alpha > 10^{-2}$ the samples produced by the Hastings algorithm were the same trivial analyses as those produced by the IO algorithm, but as α was reduced below this $\mathbf{t}$ began to exhibit nontrivial structure. We evaluated the quality of the segmentations in the morphological analyses $\mathbf{t}$ in terms of unlabeled precision, recall, f-score and exact match (the fraction of words correctly segmented into morphemes; we ignored morpheme labels because the manual morphological analyses contain many morpheme labels that we did not include in G). Figure 3 contains a plot of how these quantities vary with α; obtaining an f-score of 0.75 and an exact word match accuracy of 0.54 at $\alpha = 10^{-5}$ (the corresponding values for the MLE $\hat{\theta}$ are both 0). Note that we obtained good results as α was varied over several orders of magnitude, so the actual value of α is not critical. Thus in this application the ability to prefer sparse grammars enables us to find linguistically meaningful analyses. This ability to find linguistically meaningful structure is relatively rare in our experience with unsupervised PCFG induction.

We also experimented with a version of IO modified to perform Bayesian MAP estimation, where the Maximization step of the IO procedure is replaced with Bayesian inference using a Dirichlet prior, i.e., where the rule probabilities $\theta^{(k)}$ at iteration k are estimated using:

$$\theta_r^{(k)} \quad \propto \quad \max(0, \mathrm{E}[f_r|\mathbf{w}, \theta^{(k-1)}] + \alpha - 1).$$

Clearly such an approach is very closely related to the Bayesian procedures presented in this article,

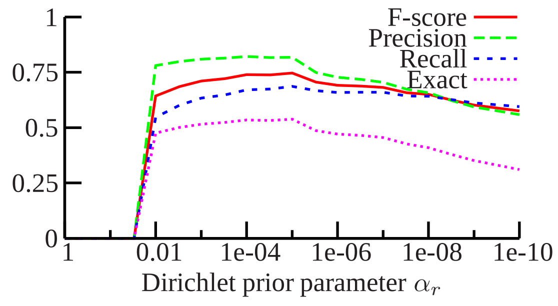


Figure 3: Accuracy of morphological segmentations of Sesotho verbs proposed by the Hastings algorithms as a function of Dirichlet prior parameter α. F-score, precision and recall are unlabeled morpheme scores, while Exact is the fraction of words correctly segmented.

and in some circumstances this may be a useful estimator. However, in our experiments with the Sesotho data above we found that for the small values of α necessary to obtain a sparse solution,the expected rule count $\mathrm{E}[f_r]$ for many rules r was less than $1 - \alpha$. Thus on the next iteration $\theta_r = 0$, resulting in there being no parse whatsoever for many of the strings in the training data. Variational Bayesian techniques offer a systematic way of dealing with these problems, but we leave this for further work.

7 Conclusion

This paper has described basic algorithms for performing Bayesian inference over PCFGs given terminal strings. We presented two Markov chain Monte Carlo algorithms (a Gibbs and a Hastings sampling algorithm) for sampling from the posterior distribution over parse trees given a corpus of their yields and a Dirichlet product prior over the production probabilities. As a component of these algorithms we described an efficient dynamic programming algorithm for sampling trees from a PCFG which is useful in its own right. We used these sampling algorithms to infer morphological analyses of Sesotho verbs given their strings (a task on which the standard Maximum Likelihood estimator returns a trivial and linguistically uninteresting solution), achieving 0.75 unlabeled morpheme f-score and 0.54 exact word match accuracy. Thus this is one of the few cases we are aware of in which a PCFG estimation procedure returns linguistically meaningful structure. We attribute this to the ability of the Bayesian prior to prefer sparse grammars.

We expect that these algorithms will be of interest to the computational linguistics community both because a Bayesian approach to PCFG estimation is more flexible than the Maximum Likelihood methods that currently dominate the field (c.f., the use of a prior as a bias towards sparse solutions), and because these techniques provide essential building blocks for more complex models.

References

Katherine Demuth. 1992. Acquisition of Sesotho. In Dan Slobin, editor, *The Cross-Linguistic Study of Language Acquisition*, volume 3, pages 557–638. Lawrence Erlbaum Associates, Hillsdale, N.J.

Ye Ding, Chi Yu Chan, and Charles E. Lawrence. 2005. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, 11:1157–1166.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Sydney, Australia. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

James E. Gentle. 2003. *Random number generation and Monte Carlo methods*. Springer, New York, 2nd edition.

Joshua Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University. available from http://research.microsoft.com/˜joshuago/.

Dan Klein and Chris Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485.

K. Lari and S.J. Young. 1990. The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(35-56).

Brian MacWhinney and Catherine Snow. 1985. The child language data exchange system. *Journal of Child Language*, 12:271–296.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia. Association for Computational Linguistics.

Worst-Case Synchronous Grammar Rules

Daniel Gildea and **Daniel Štefankovič**
Computer Science Dept.
University of Rochester
Rochester, NY 14627

Abstract

We relate the problem of finding the best application of a Synchronous Context-Free Grammar (SCFG) rule during parsing to a Markov Random Field. This representation allows us to use the theory of expander graphs to show that the complexity of SCFG parsing of an input sentence of length N is $\Omega(N^{cn})$, for a grammar with maximum rule length n and some constant c. This improves on the previous best result of $\Omega(N^{c\sqrt{n}})$.

1 Introduction

Recent interest in syntax-based methods for statistical machine translation has lead to work in parsing algorithms for synchronous context-free grammars (SCFGs). Generally, parsing complexity depends on the length of the longest rule in the grammar, but the exact nature of this relationship has only recently begun to be explored. It has been known since the early days of automata theory (Aho and Ullman, 1972) that the languages of string pairs generated by a synchronous grammar can be arranged in an infinite hierarchy, with each rule size ≥ 4 producing languages not possible with grammars restricted to smaller rules. For any grammar with maximum rule size n, a fairly straightforward dynamic programming strategy yields an $O(N^{n+4})$ algorithm for parsing sentences of length N. However, this is often not the best achievable complexity, and the exact bounds of the best possible algorithms are not known. Satta and Peserico (2005) showed that a permutation can be defined for any length n such that tabular parsing strategies must take at least $\Omega(N^{c\sqrt{n}})$, that is, the exponent of the algorithm is proportional to the square root of the rule length. In this paper, we improve this result, showing that in the worst case the exponent grows linearly with the rule length. Using a probabilistic argument, we show that the number of easily parsable permutations grows slowly enough that most permutations must be difficult, where by difficult we mean that the exponent in the complexity is greater than a constant factor times the rule length. Thus, not only do there exist permutations that have complexity higher than the square root case of Satta and Peserico (2005), but in fact the probability that a randomly chosen permutation will have higher complexity approaches one as the rule length grows.

Our approach is to first relate the problem of finding an efficient parsing algorithm to finding the *treewidth* of a graph derived from the SCFG rule's permutation. We then show that this class of graphs are *expander graphs*, which in turn means that the treewidth grows linearly with the graph size.

2 Synchronous Parsing Strategies

We write SCFG rules as productions with one lefthand side nonterminal and two righthand side strings. Nonterminals in the two strings are linked with superscript indices; symbols with the same index must be further rewritten synchronously. For example,

$$X \rightarrow A^{(1)}\, B^{(2)}\, C^{(3)}\, D^{(4)},\; A^{(1)}\, B^{(2)}\, C^{(3)}\, D^{(4)} \tag{1}$$

is a rule with four children and no reordering, while

$$X \rightarrow A^{(1)}\, B^{(2)}\, C^{(3)}\, D^{(4)},\; B^{(2)}\, D^{(4)}\, A^{(1)}\, C^{(3)} \tag{2}$$

Proceedings of NAACL HLT 2007, pages 147–154,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

Algorithm 1 BottomUpParser(grammar G, input strings $\mathbf{e}$, $\mathbf{f}$)

```
for $x_0, x_n$ such that $1 < x_0 < x_n < |\mathbf{e}|$ in increasing order of $x_n - x_0$ do
    for $y_0, y_n$ such that $1 < y_0 < y_n < |\mathbf{f}|$ in increasing order of $y_n - y_0$ do
        for Rules $R$ of form $X \rightarrow X_1^{(1)}...X_n^{(n)},\ X_{\pi(1)}^{(\pi(1))}...X_{\pi(n)}^{(\pi(n))}$ in $G$ do
            $p = P(R) \max_{\substack{x_1..x_{n-1}\\ y_1..y_{n-1}}} \prod_i \delta(X_i, x_{i-1}, x_i, y_{\pi(i)-1}, y_{\pi(i)})$
            $\delta(X, x_0, x_n, y_0, y_n) = \max\{\delta(X, x_0, x_n, y_0, y_n), p\}$
        end for
    end for
end for
```

expresses a more complex reordering. In general, we can take indices in the first grammar dimension to be consecutive, and associate a permutation π with the second dimension. If we use X_i for $0 \leq i \leq n$ as a set of variables over nonterminal symbols (for example, X_1 and X_2 may both stand for nonterminal A), we can write rules in the general form:

$$X_0 \rightarrow X_1^{(1)}...X_n^{(n)},\ X_{\pi(1)}^{(\pi(1))}...X_{\pi(n)}^{(\pi(n))}$$

Grammar rules also contain terminal symbols, but as their position does not affect parsing complexity, we focus on nonterminals and their associated permutation π in the remainder of the paper. In a probabilistic grammar, each rule R has an associated probability $P(R)$. The synchronous parsing problem consists of finding the tree covering both strings having the maximum product of rule probabilities.[1]

We assume synchronous parsing is done by storing a dynamic programming table of recognized nonterminals, as outlined in Algorithm 1. We refer to a dynamic programming item for a given nonterminal with specified boundaries in each language as a *cell*. The algorithm computes cells by maximizing over *boundary variables* x_i and y_i, which range over positions in the two input strings, and specify beginning and end points for the SCFG rule's child nonterminals.

The maximization in the inner loop of Algorithm 1 is the most expensive part of the procedure, as it would take $O(N^{2n-2})$ with exhaustive search; making this step more efficient is our focus in this paper. The maximization can be done with further dynamic programming, storing partial results which contain some subset of an SCFG rule's righthand side nonterminals that have been recognized. A parsing strategy for a specific SCFG rule consists of an order in which these subsets should be combined, until all the rule's children have been recognized. The complexity of an individual parsing step depends on the number of free boundary variables, each of which can take $O(N)$ values. It is often helpful to visualize parsing strategies on the *permutation matrix* corresponding to a rule's permutation π. Figure 1 shows the permutation matrix of rule (2) with a three-step parsing strategy. Each panel shows one combination step along with the projections of the partial results in each dimension; the endpoints of these projections correspond to free boundary variables. The second step has the highest number of distinct endpoints, five in the vertical dimension and three horizontally, meaning parsing can be done in time $O(N^8)$.

As an example of the impact that the choice of parsing strategy can make, Figure 2 shows a permutation for which a clever ordering of partial results enables parsing in time $O(N^{10})$ in the length of the input strings. Permutations having this pattern of diagonal stripes can be parsed using this strategy in time $O(N^{10})$ regardless of the length n of the SCFG rule, whereas a naïve strategy proceeding from left to right in either input string would take time $O(N^{n+3})$.

2.1 Markov Random Fields for Cells

In this section, we connect the maximization of probabilities for a cell to the Markov Random Field

[1]We describe our methods in terms of the Viterbi algorithm (using the max-product semiring), but they also apply to non-probabilistic parsing (boolean semiring), language modeling (sum-product semiring), and Expectation Maximization (with inside and outside passes).

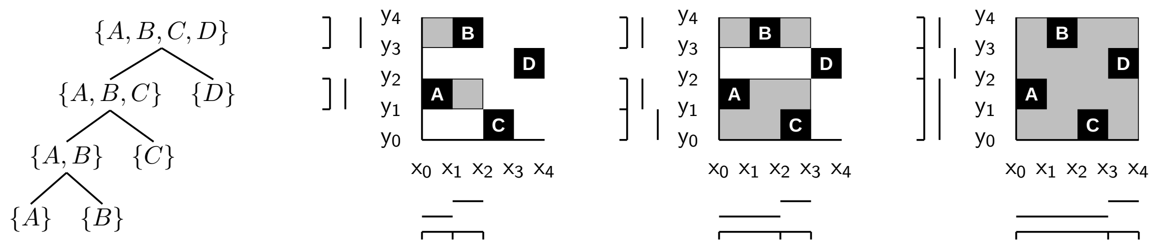


Figure 1: The tree on the left defines a three-step parsing strategy for rule (2). In each step, the two subsets of nonterminals in the inner marked spans are combined into a new chart item with the outer spans. The intersection of the outer spans, shaded, has now been processed. Tic marks indicate distinct endpoints of the spans being combined, corresponding to the free boundary variables.

(MRF) representation, which will later allow us to use algorithms and complexity results based on the graphical structure of MRFs. A Markov Random Field is defined as a probability distribution[2] over a set of variables $\mathbf{x}$ that can be written as a product of *factors* f_i that are functions of various subsets $\mathbf{x}_i$ of $\mathbf{x}$. The probability of an SCFG rule instance computed by Algorithm 1 can be written in this functional form:

$$\delta^R(\mathbf{x}) = P(R) \prod_i f_i(\mathbf{x}_i)$$

where

$$\mathbf{x} = \{x_i, y_i\} \quad \text{for } 0 \le i \le n$$

$$\mathbf{x}_i = \{x_{i-1}, x_i, y_{\pi(i)-1}, y_{\pi(i)}\}$$

and the MRF has one factor f_i for each child nonterminal X_i in the grammar rule R. The factor's value is the probability of the child nonterminal, which can be expressed as a function of its four boundaries:

$$f_i(\mathbf{x}_i) \quad = \quad \delta(X_i, x_{i-1}, x_i, y_{\pi(i)-1}, y_{\pi(i)})$$

For reasons that are explained in the following section, we augment our Markov Random Fields with a dummy factor for the completed parent nonterminal's chart item. Thus there is one dummy factor d for each grammar rule:

$$d(x_0, x_n, y_0, y_n) = 1$$

expressed as a function of the four *outer boundary variables* of the completed rule, but with a constant value of 1 so as not to change the probabilities computed.

[2]In our case unnormalized.

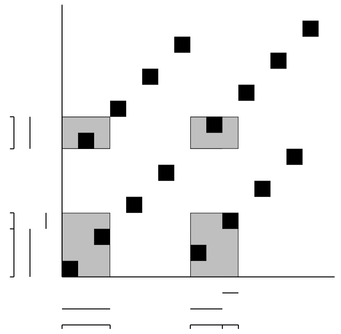


Figure 2: A parsing strategy maintaining two spans in each dimension is $O(N^{10})$ for any length permutation of this general form.

Thus an SCFG rule with n child nonterminals always results in a Markov Random Field with $2n+2$ variables and $n+1$ factors, with each factor a function of exactly four variables.

Markov Random Fields are often represented as graphs. A *factor graph* representation has a node for each variable and factor, with an edge connecting each factor to the variables it depends on. An example for rule (2) is shown in Figure 3, with round nodes for variables, square nodes for factors, and a diamond for the special dummy factor.

2.2 Junction Trees

Efficient computation on Markov Random Fields is performed by first transforming the MRF into a *junction tree* (Jensen et al., 1990; Shafer and Shenoy, 1990), and then applying the standard

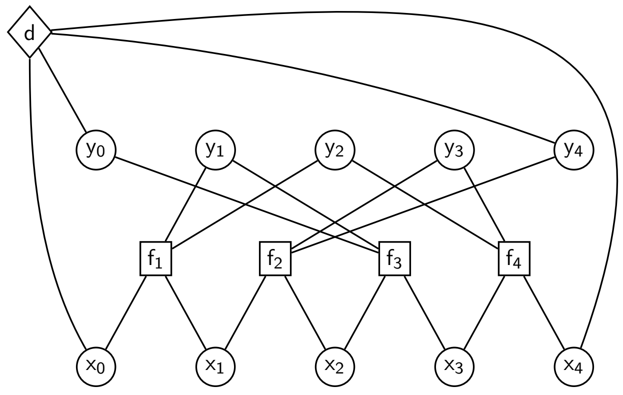


Figure 3: Markov Random Field for rule (2).

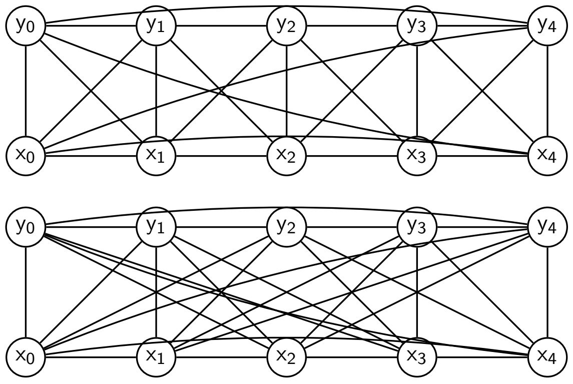


Figure 4: The graphs resulting from connecting all interacting variables for the identity permutation $(1, 2, 3, 4)$ (top) and the $(2, 4, 1, 3)$ permutation of rule (2) (bottom).

message-passing algorithm for graphical models over this tree structure. The complexity of the message passing algorithm depends on the structure of the junction tree, which in turn depends on the graph structure of the original MRF.

A junction tree can be constructed from a Markov Random Field by the following three steps:

- Connect all variable nodes that share a factor, and remove factor nodes. This results in the graphs shown in Figure 4.
- Choose a *triangulation* of the resulting graph, by adding chords to any cycle of length greater than three.
- Decompose the triangulated graph into a tree of cliques.

We call nodes in the resulting tree, corresponding to cliques in the triangulated graph, *clusters*. Each cluster has a *potential function*, which is a function of the variables in the cluster. For each factor in the original MRF, the junction tree will have at least one cluster containing all of the variables on which the factor is defined. Each factor is associated with one such cluster, and the cluster's potential function is set to be the product of its factors, for all combinations of variable values. Triangulation ensures that the resulting tree satisfies the *junction tree property*, which states that for any two clusters containing the same variable x, all nodes on the path connecting the clusters also contain x. A junction tree derived from the MRF of Figure 3 is shown in Figure 5.

The message-passing algorithm for graphical models can be applied to the junction tree. The algorithm works from the leaves of the tree inward, alternately multiplying in potential functions and maximizing over variables that are no longer needed, effectively distributing the $\max$ and product operators so as to minimize the interaction between variables. The complexity of the message-passing is $O(nN^k)$, where the junction tree contain $O(n)$ clusters, k is the maximum cluster size, and each variable in the cluster can take N values.

However, the standard algorithm assumes that the factor functions are predefined as part of the input. In our case, however, the factor functions themselves depend on message-passing calculations from other grammar rules:

$$\begin{aligned} f_i(\mathbf{x}_i) &= \delta(X_i, x_{i-1}, x_i, y_{\pi(i)-1}, y_{\pi(i)}) \\ &= \max_{R': X_i \to \alpha, \beta} P(R') \max_{\substack{\mathbf{x}': \\ x'_0 = x_{i-1}, x'_{n'} = x_i \\ y'_0 = y_{\pi(i-1)}, y'_{n'} = y_{\pi(i)}}} \delta^{R'}(\mathbf{x}') \end{aligned} \quad (3)$$

We must modify the standard algorithm in order to interleave computation among the junction trees corresponding to the various rules in the grammar, using the bottom-up ordering of computation from Algorithm 1. Where, in the standard algorithm, each message contains a complete table for all assignments to its variables, we break these into a separate message for each individual assignment of variables. The overall complexity is unchanged, because each assignment to all variables in each cluster is still considered only once.

The dummy factor d ensures that every junction

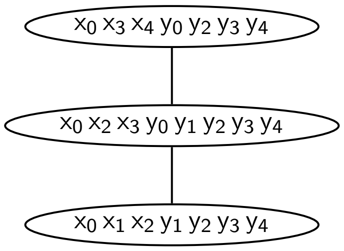


Figure 5: Junction tree for rule (2).

tree we derive from an SCFG rule has a cluster containing all four outer boundary variables, allowing efficient lookup of the inner maximization in (3). Because the outer boundary variables need not appear throughout the junction tree, this technique allows reuse of some partial results across different outer boundaries. As an example, consider message passing on the junction tree of shown in Figure 5, which corresponds to the parsing strategy of Figure 1. Only the final step involves all four boundaries of the complete cell, but the most complex step is the second, with a total of eight boundaries. This efficient reuse would not be achieved by applying the junction tree technique directly to the maximization operator in Algorithm 1, because we would be fixing the outer boundaries and computing the junction tree only over the inner boundaries.

3 Treewidth and Tabular Parsing

The complexity of the message passing algorithm over an MRF's junction tree is determined by the *treewidth* of the MRF. In this section we show that, because parsing strategies are in direct correspondence with valid junction trees, we can use treewidth to analyze the complexity of a grammar rule.

We define a tabular parsing strategy as any dynamic programming algorithm that stores partial results corresponding to subsets of a rule's child nonterminals. Such a strategy can be represented as a recursive partition of child nonterminals, as shown in Figure 1(left). We show below that a recursive partition of children having maximum complexity k at any step can be converted into a junction tree having k as the maximum cluster size. This implies that finding the optimal junction tree will give a parsing strategy at least as good as the strategy of the optimal recursive partition.

A recursive partition of child nonterminals can be converted into a junction tree as follows:

- For each leaf of the recursive partition, which represents a single child nonterminal i, create a leaf in the junction tree with the cluster $(x_{i-1}, x_i, y_{\pi(i)-1}, y_{\pi(i)})$ and the potential function $f_i(x_{i-1}, x_i, y_{\pi(i)-1}, y_{\pi(i)})$.
- For each internal node in the recursive partition, create a corresponding node in the junction tree.
- Add each variable x_i to all nodes in the junction tree on the path from the node for child nonterminal $i-1$ to the node for child nonterminal i. Similarly, add each variable $y_{\pi(i)}$ to all nodes in the junction tree on the path from the node for child nonterminal $\pi(i)-1$ to the node for child nonterminal $\pi(i)$.

Because each variable appears as an argument of only two factors, the junction tree nodes in which it is present form a linear path from one leaf of the tree to another. Since each variable is associated only with nodes on one path through the tree, the resulting tree will satisfy the junction tree property. The tree structure of the original recursive partition implies that the variable rises from two leaf nodes to the lowest common ancestor of both leaves, and is not contained in any higher nodes. Thus each node in the junction tree contains variables corresponding to the set of endpoints of the spans defined by the two subsets corresponding to its two children. The number of variables at each node in the junction tree is identical to the number of free endpoints at the corresponding combination in the recursive partition.

Because each recursive partition corresponds to a junction tree with the same complexity, finding the best recursive partition reduces to finding the junction tree with the best complexity, i.e., the smallest maximum cluster size.

Finding the junction tree with the smallest cluster size is equivalent to finding the input graph's *treewidth*, the smallest k such that the graph can be embedded in a k-tree. In general, this problem was shown to be NP-complete by Arnborg et al. (1987). However, because the treewidth of a given rule lower bounds the complexity of its tabular parsing strategies, parsing complexity for general rules can be bounded with treewidth results for worst-case rules, without explicitly identifying the worst-case permutations.

4 Treewidth Grows Linearly

In this section, we show that the treewidth of the graphs corresponding to worst-case permutations growths linearly with the permutation's length. Our strategy is as follows:

1. Define a 3-regular graph for an input permutation consisting of a subset of edges from the original graph.
2. Show that the edge-expansion of the 3-regular graph grows linearly for randomly chosen permutations.
3. Use edge-expansion to bound the spectral gap.
4. Use spectral gap to bound treewidth.

For the first step, we define $H = (V, E)$ as a random 3-regular graph on $2n$ vertices obtained as follows. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be cycles, each on a separate set of n vertices. These two cycles correspond to the edges (x_i, x_{i+1}) and (y_i, y_{i+1}) in the graphs of the type shown in Figure 4. Let M be a random perfect matching between V_1 and V_2. The matching represents the edges $(x_i, y_{\pi(i)})$ produced from the input permutation π. Let H be the union of G_1, G_2, and M. While H contains only some of the edges in the graphs defined in the previous section, removing edges cannot increase the treewidth.

For the second step of the proof, we use a probabilistic argument detailed in the next subsection.

For the third step, we will use the following connection between the edge-expansion and the eigenvalue gap (Alon and Milman, 1985; Tanner, 1984).

Lemma 4.1 *Let G be a k-regular graph. Let λ_2 be the second largest eigenvalue of G. Let $h(G)$ be the edge-expansion of G. Then*

$$k - \lambda_2 \geq \frac{h(G)^2}{2k}.$$

Finally, for the fourth step, we use a relation between the eigenvalue gap and treewidth for regular graphs shown by Chandran and Subramanian (2003).

Lemma 4.2 *Let G be a k-regular graph. Let n be the number of vertices of G. Let λ_2 be the second largest eigenvalue of G. Then*

$$\operatorname{tw}(G) \geq \left\lfloor \frac{n}{4k} (k - \lambda_2) \right\rfloor - 1$$

Note that in our setting $k = 3$. In order to use Lemma 4.2 we will need to give a lower bound on the eigenvalue gap $k - \lambda_2$ of G.

4.1 Edge Expansion

The *edge-expansion* of a set of vertices T is the ratio of the number of edges connecting vertices in T to the rest of the graph, divided by the number of vertices in T,

$$\frac{|E(T, V - T)|}{|T|}$$

where we assume that $|T| \leq |V|/2$. The edge expansion of a graph is the minimum edge expansion of any subset of vertices:

$$h(G) = \min_{T \subseteq V} \frac{|E(T, V - T)|}{\min\{|T|, |V - T|\}}.$$

Intuitively, if all subsets of vertices are highly connected to the remainder of the graph, there is no way to decompose the graph into minimally interacting subgraphs, and thus no way to decompose the dynamic programming problem of parsing into smaller pieces.

Let $\binom{n}{k}$ be the standard binomial coefficient, and for $\alpha \in \mathbb{R}$, let

$$\binom{n}{\leq \alpha} = \sum_{k=0}^{\lfloor \alpha \rfloor} \binom{n}{k}.$$

We will use the following standard inequality valid for $0 \leq \alpha \leq n$:

$$\binom{n}{\leq \alpha} \leq \left(\frac{ne}{\alpha}\right)^{\alpha} \tag{4}$$

Lemma 4.3 *With probability at least 0.98 the graph H has edge-expansion at least $1/50$.*

Proof :
Let $\varepsilon = 1/50$. Assume that $T \subseteq V$ is a set with a small edge-expansion, i. e.,

$$|E(T, V - T)| \leq \varepsilon |T|, \tag{5}$$

and $|T| \leq |V|/2 = n$. Let $T_i = T \cap V_i$ and let $t_i = |T_i|$, for $i = 1, 2$. We will w.l.o.g. assume $t_1 \leq t_2$. We will denote as ℓ_i the number of spans of consecutive vertices from E_i contained in T. Thus $2\ell_i = |E(T_i, V_i - T_i)|$, for $i = 1, 2$. The spans counted by ℓ_1 and ℓ_2 correspond to continuous spans counted in computing the complexity of a chart parsing operation. However, unlike in the diagrams in the earlier part of this paper, in our graph theoretic argument there is no requirement that T select only corresponding pairs of vertices from V_1 and V_2.

There are at least $2(\ell_1+\ell_2)+t_2-t_1$ edges between T and $V - T$. This is because there are $2\ell_i$ edges within V_i at the left and right boundaries of the ℓ_i spans, and at least $t_2 - t_1$ edges connecting the extra vertices from T_2 that have no matching vertex in T_1. Thus from assumption (5) we have

$$t_2 - t_1 \leq \varepsilon(t_1 + t_2)$$

which in turn implies

$$t_1 \leq t_2 \leq \frac{1+\varepsilon}{1-\varepsilon}\, t_1. \tag{6}$$

Similarly, using (6), we have

$$\ell_1 + \ell_2 \leq \frac{\varepsilon}{2}\,(t_1 + t_2) \leq \frac{\varepsilon}{1-\varepsilon}\, t_1. \tag{7}$$

That is, for T to have small edge expansion, the vertices in T_1 and T_2 must be collected into a small number of spans ℓ_1 and ℓ_2. This limit on the number of spans allows us to limit the number of ways of choosing T_1 and T_2. Suppose that t_1 is given. Any pair T_1, T_2 is determined by the edges in $E(T_1, V_1 - T_1)$, and $E(T_2, V_2 - T_2)$, and two bits (corresponding to the possible "swaps" of T_i with $V_i - T_i$). Note that we can choose at most $2\ell_1 + 2\ell_2 \leq t_1 \cdot 2\varepsilon/(1-\varepsilon)$ edges in total. Thus the number of choices of T_1 and T_2 is bounded above by

$$4 \cdot \binom{2n}{\leq \frac{2\varepsilon}{1-\varepsilon}\, t_1}. \tag{8}$$

For a given choice of T_1 and T_2, for T to have small edge expansion, there must also not be too many edges that connect T_1 to vertices in $V_2 - T_2$. Let k be the number of edges between T_1 and T_2. There are at least $t_1 + t_2 - 2k$ edges between T and $V - T$ and from assumption (5) we have

$$t_1 + t_2 - 2k \leq \varepsilon(t_1 + t_2)$$

Thus

$$k \geq (1-\varepsilon)\frac{t_1 + t_2}{2} \geq (1-\varepsilon)t_1. \tag{9}$$

The probability that there are $\geq (1-\varepsilon)t_1$ edges between T_1 and T_2 is bounded by

$$\binom{t_1}{\leq \varepsilon t_1} \left(\frac{t_2}{n}\right)^{(1-\varepsilon)t_1}$$

where the first term selects vertices in T_1 connected to T_2, and the second term upper bounds the probability that the selected vertices are indeed connected to T_2. Using 6, we obtain a bound in terms of t_1 alone:

$$\binom{t_1}{\leq \varepsilon t_1} \left(\frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{t_1}{n}\right)^{(1-\varepsilon)t_1}, \tag{10}$$

Combining the number of ways of choosing T_1 and T_2 (8) with the bound on the probability that the edges M from the input permutation connect almost all the vertices in T_1 to vertices from T_2 (10), and using the union bound over values of t_1, we obtain that the probability p that there exists $T \subseteq V$ with edge-expansion less than ε is bounded by:

$$2 \sum_{t_1=0}^{\lfloor n/2 \rfloor} 4 \cdot \binom{2n}{\leq \frac{2\varepsilon}{1-\varepsilon}\, t_1} \binom{t_1}{\leq \varepsilon t_1} \left(\frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{t_1}{n}\right)^{(1-\varepsilon)t_1} \tag{11}$$

where the factor of 2 is due to the assumption $t_1 \leq t_2$.

The graph H is connected and hence T has at least one out-going edge. Therefore if $t_1 + t_2 \leq 1/\varepsilon$, the edge-expansion of T is at least ε. Thus a set with edge-expansion less than ε must have $t_1 + t_2 \geq 1/\varepsilon$, which, by (6), implies $t_1 \geq (1-\varepsilon)/(2\varepsilon)$. Thus the sum in (11) can be taken for t from $\lceil (1-\varepsilon)/(2\varepsilon) \rceil$

to $\lfloor n/2 \rfloor$. Using (4) we obtain

$$p \leq 8 \sum_{t_1=\lceil \frac{1-\varepsilon}{2\varepsilon} \rceil}^{\lfloor n/2 \rfloor} \left[\left(\frac{2n\mathrm{e}}{\frac{2\varepsilon}{1-\varepsilon} t_1} \right)^{\frac{2\varepsilon}{1-\varepsilon} t_1} \left(\frac{t_1 \mathrm{e}}{\varepsilon t_1} \right)^{\varepsilon t_1} \left(\frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{t_1}{n} \right)^{(1-\varepsilon) t_1} \right] =$$
$$8 \sum_{t_1=\lceil \frac{1-\varepsilon}{2\varepsilon} \rceil}^{\lfloor n/2 \rfloor} \left(\left(\frac{\mathrm{e}(1-\varepsilon)}{\varepsilon} \right)^{\frac{2\varepsilon}{1-\varepsilon}} \left(\frac{\mathrm{e}}{\varepsilon} \right)^{\varepsilon} \left(\frac{1+\varepsilon}{1-\varepsilon} \right)^{1-\varepsilon} \left(\frac{t_1}{n} \right)^{1-\varepsilon-\frac{2\varepsilon}{1-\varepsilon}} \right)^{t_1}. \tag{12}$$

We will use $t_1/n \leq 1/2$ and plug $\varepsilon = 1/50$ into (12). We obtain

$$p \leq 8 \sum_{t_1=25}^{\infty} 0.74^{t_1} \leq 0.02.$$

■

While this constant bound on p is sufficient for our main complexity result, it can further be shown that p approaches zero as n increases, from the fact that the geometric sum in (12) converges, and each term for fixed t_1 goes to zero as n grows.

This completes the second step of the proof as outlined at the beginning of this section. The constant bound on the edge expansion implies a constant bound on the eigenvalue gap (Lemma 4.1), which in turn implies an $\Omega(n)$ bound on treewidth (Lemma 4.2), yielding:

Theorem 4.4 *Tabular parsing strategies for Synchronous Context-Free Grammars containing rules with all permutations of length n require time $\Omega(N^{cn})$ in the input string length N for some constant c.*

We have shown our result without explicitly constructing a difficult permutation, but we close with one example. The zero-based permutations of length p, where p is prime, $\pi(i) = i^{-1} \mod p$ for $0 < i < p$, and $\pi(0) = 0$, provide a known family of expander graphs (see Hoory et al. (2006)).

5 Conclusion

We have shown in the exponent in the complexity of polynomial-time parsing algorithms for synchronous context-free grammars grows linearly with the length of the grammar rules. While it is very expensive computationally to test whether a specified permutation has a parsing algorithm of a certain complexity, it turns out that randomly chosen permutations are difficult with high probability.

Acknowledgments This work was supported by NSF grants IIS-0546554, IIS-0428020, and IIS-0325646.

References

Albert V. Aho and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

N. Alon and V.D. Milman. 1985. λ_1, isoperimetric inequalities for graphs and superconcentrators. *J. of Combinatorial Theory, Ser. B*, 38:73–88.

Stefen Arnborg, Derek G. Corneil, and Andrzej Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8:277–284, April.

L.S. Chandran and C.R. Subramanian. 2003. A spectral lower bound for the treewidth of a graph and its consequences. *Information Processing Letters*, 87:195–200.

Shlomo Hoory, Nathan Linial, and Avi Wigderson. 2006. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561.

Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.

Giorgio Satta and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of HLT/EMNLP*, pages 803–810, Vancouver, Canada, October.

G. Shafer and P. Shenoy. 1990. Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–353.

R.M. Tanner. 1984. Explicit construction of concentrators from generalized n-gons. *J. Algebraic Discrete Methods*, 5:287–294.

High-Performance, Language-Independent Morphological Segmentation

Sajib Dasgupta and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{sajib,vince}@hlt.utdallas.edu

Abstract

This paper introduces an unsupervised morphological segmentation algorithm that shows robust performance for four languages with different levels of morphological complexity. In particular, our algorithm outperforms Goldsmith's Linguistica and Creutz and Lagus's Morphessor for English and Bengali, and achieves performance that is comparable to the best results for all three PASCAL evaluation datasets. Improvements arise from (1) the use of relative corpus frequency and suffix level similarity for detecting incorrect morpheme attachments and (2) the induction of orthographic rules and allomorphs for segmenting words where roots exhibit spelling changes during morpheme attachments.

1 Introduction

Morphological analysis is the task of segmenting a word into *morphemes*, the smallest meaning-bearing elements of natural languages. Though very successful, *knowledge-based* morphological analyzers operate by relying on manually designed segmentation heuristics (e.g. Koskenniemi (1983)), which require a lot of linguistic expertise and are time-consuming to construct. As a result, research in morphological analysis has exhibited a shift to *unsupervised* approaches, in which a word is typically segmented based on morphemes that are automatically induced from an unannotated corpus. Unsupervised approaches have achieved considerable success for English and many European languages (e.g. Goldsmith (2001), Schone and Jurafsky (2001), Freitag (2005)). The recent PASCAL Challenge on *Unsupervised Segmentation of Words into Morphemes*[1] has further intensified interest in this problem, selecting as target languages English as well as two highly agglutinative languages, Turkish and Finnish. However, the evaluation of the Challenge reveals that (1) the success of existing unsupervised morphological parsers does not carry over to the two agglutinative languages, and (2) no segmentation algorithm achieves good performance for all three languages.

Motivated by these state-of-the-art results, our goal in this paper is to develop an *unsupervised* morphological segmentation algorithm that can *work well across different languages*. With this goal in mind, we evaluate our algorithm on four languages with different levels of morphological complexity, namely English, Turkish, Finnish and Bengali. It is worth noting that Bengali is an under-investigated Indo-Aryan language that is highly inflectional and lies between English and Turkish/Finnish in terms of morphological complexity. Experimental results demonstrate the robustness of our algorithm across languages: it not only outperforms Goldsmith's (2001) Linguistica and Creutz and Lagus's (2005) Morphessor for English and Bengali, but also compares favorably to the best-performing PASCAL morphological parsers when evaluated on all three datasets in the Challenge.

The performance improvements of our segmentation algorithm over existing morphological analyzers can be attributed to our extending Keshava and Pitler's (2006) segmentation method, the best performer for English in the aforementioned

[1] http://www.cis.hut.fi/morphochallenge2005/

PASCAL Challenge, with the capability of handling two under-investigated problems:

Detecting incorrect attachments. Many existing morphological parsers incorrectly segment "candidate" as "candid"+"ate", since they fail to identify that the morpheme "ate" should *not* attach to the word "candid". Schone and Jurafsky's (2001) work represents one of the few attempts to address this inappropriate morpheme attachment problem, introducing a method that exploits the semantic relatedness between word pairs. In contrast, we propose two arguably simpler, yet effective techniques that rely on *relative corpus frequency* and *suffix level similarity* to solve the problem.

Inducing orthographic rules and allomorphs. One problem with Keshava and Pitler's algorithm is that it fails to segment words where the roots exhibit spelling changes during attachment to morphemes (e.g. "denial" = "deny"+"al"). To address this problem, we automatically acquire allomorphs and orthographic change rules from an unannotated corpus. These rules also allow us to output the actual segmentation of the words that exhibit spelling changes during morpheme attachment, thus avoiding the segmentation of "denial" as "deni"+"al", as is typically done in existing morphological parsers.

In addition to addressing the aforementioned problems, our segmentation algorithm has two appealing features. First, it can segment words with any number of morphemes, whereas many analyzers can only be applied to words with one root and one suffix (e.g. DéJean (1998), Snover and Brent (2001)). Second, it exhibits robust performance even when inducing morphemes from a very large vocabulary, whereas Goldsmith's (2001) and Freitag's (2005) morphological analyzers perform well only when a small vocabulary is employed, showing deteriorating performance as the vocabulary size increases.

The rest of this paper is organized as follows. Section 2 presents related work on unsupervised morphological analysis. In Section 3, we describe our basic morpheme induction algorithm. We then show how to exploit the induced morphemes to (1) detect incorrect attachments by using relative corpus frequency (Section 4) and suffix level similarity (Section 5) and (2) induce orthographic rules and allomorphs (Section 6). Section 7 describes our algorithm for segmenting a word using the induced morphemes. We present evaluation results in Section 8 and conclude in Section 9.

2 Related Work

As mentioned in the introduction, the problem of unsupervised morphological learning has been extensively studied for English and many other European languages. In this section, we will give an overview of the related work on this problem.

Harris (1955) develops a strategy for identifying morpheme boundaries that checks whether the number of different letters following a sequence of letters exceeds some given threshold. DéJean (1998) improves Harris's segmentation algorithm by first inducing a list of 100 most frequent morphemes and then using those morphemes for word segmentation. The aforementioned PASCAL Challenge on Unsupervised Word Segmentation has undoubtedly intensified interest in this problem. Among the participating groups, Keshava and Pitler's (2006) segmentation algorithm combines the ideas of DéJean and Harris and achieves the best result for the English dataset, but it only offers mediocre performance for Finnish and Turkish.

There is another class of unsupervised morphological learning algorithms whose design is driven by the Minimum Description Length (MDL) principle. Specifically, EM is used to iteratively segment a list of words using some predefined heuristics until the length of the morphological grammar converges to a minimum. Brent et al. (1995) are the first to introduce an information-theoretic notion of compression to represent the MDL framework. Goldsmith (2001) also adopts the MDL approach, providing a new compression system that incorporates signatures when measuring the length of the morphological grammar. Creutz (2003) proposes a probabilistic maximum *a posteriori* formulation that uses prior distributions of morpheme length and frequency to measure the goodness of an induced morpheme, achieving better results for Finnish but worse results for English in comparison to Goldsmith's Linguistica.

3 The Basic Morpheme Induction Algorithm

Our unsupervised segmentation algorithm is composed of two steps: (1) inducing *prefixes*, *suffixes* and *roots* from a vocabulary that consists of words taken from a large corpus, and (2) segmenting a word using these induced morphemes. This section describes our *basic* morpheme induction method.

3.1 Extracting a List of Candidate Affixes

The first step of our morpheme induction method involves extracting a list of candidate prefixes and suffixes. We rely on a fairly simple idea originally proposed by Keshava and Pitler (2006) for extracting candidate affixes. Assume that α and β are two character sequences and $\alpha\beta$ is the concatenation of α and β. If $\alpha\beta$ and α are both found in the vocabulary, then we extract β as a candidate suffix. Similarly, if $\alpha\beta$ and β are both found in the vocabulary, then we extract α as a candidate prefix.

The above affix induction method is arguably overly simplistic and therefore can generate many spurious affixes. To filter spurious affixes, we (1) score each affix by multiplying its *frequency* (i.e. the number of distinct words to which each affix attaches) and its *length*[2], and then (2) retain only the *K* top-scoring affixes, where *K* is set differently for prefixes and suffixes. The value of *K* is somewhat dependent on the vocabulary size, as the affixes in a larger vocabulary system are generated from a larger number of words. For example, we set the thresholds to 70 for prefixes and 50 for suffixes for English; on the other hand, since the Finnish vocabulary is almost six times larger than that of English, we set the corresponding thresholds to be approximately six times larger (400 and 300 for prefixes and suffixes respectively).[3]

3.2 Detecting Composite Suffixes

Next, we detect and remove *composite suffixes* (i.e. suffixes that are formed by combining multiple suffixes [e.g. "ers" = "er"+"s"]) from our induced suffix list, because their presence can lead to under-segmentation of words (e.g. "walkers", whose correct segmentation is "walk"+"er"+"s", will be erroneously segmented as "walk"+"ers"). Composite suffix detection is a particularly important problem for languages like Bengali in which composite suffixes are abundant (see Dasgupta and Ng (2007)). Note, however, that simple concatenation of multiple suffixes does not always produce a composite suffix. For example, "ent", "en" and "t" all are valid suffixes in English, but "ent" is not a composite suffix. Hence, we need a more sophisticated method for composite suffix detection.

Our detection method is motivated by the following observation: if *xy* is a composite suffix and a word *w* combines with *xy*, then it is highly likely that *w* will also combine with its first component suffix *x*. Note that this property does not hold for non-composite suffixes. For instance, words that combine with the non-composite suffix "ent" (e.g. "absorb") do not combine with its first component suffix "en". Consequently, given two suffixes *x* and *y*, our method posits *xy* as a composite suffix if *xy* and *x* are similar in terms of the words to which they attach. Specifically, we consider *xy* and *x* to be similar if their similarity value as computed by the formula below is greater than 0.6:

$$\mathit{Similarity}\,(xy, x) = P(x \mid xy) = \frac{|W'|}{|W|},$$

where $|W'|$ is the number of distinct words that combine with both *xy* and *x*, and $|W|$ is the number of distinct words that combine with *xy*.

3.3 Extracting a List of Candidate Roots

Finally, we extract a list of candidate roots using the induced list of affixes as follows. For each word, *w*, in the vocabulary, we check whether *w* is *divisible*, i.e. whether *w* can be segmented as *r*+*x* or *p*+*r*, where *p* is an induced prefix, *x* is an induced suffix, and *r* is a word in the vocabulary. We then add *w* to the root list if it is not divisible. Note, however, that the resulting root list may contain *compound* words (i.e. words with multiple roots). Hence, we make another pass over our root list to remove any word that is a concatenation of multiple words in the vocabulary.

4 Detecting Incorrect Attachments Using Relative Frequency

Our induced root list is not perfect: many correct roots are missing due to over-segmentation. For example, since "candidate" and "candid" are in the vocabulary and "ate" is an induced suffix, our root induction method will incorrectly segment "candidate" as "candid"+"ate"; as a result, it does not consider "candidate" as a root. So, to improve the root induction method, we need to determine that the attachment of the morpheme "ate" to the root word "candid" is incorrect. In this section, we propose a simple yet novel idea of using relative cor-

[2] The dependence on frequency and length is motivated by the observation that less-frequent and shorter affixes (especially those of length 1) are more likely to be erroneous (see Goldsmith (2001)).

[3] Since this method for setting our vocabulary-dependent thresholds is fairly simple, the use of these thresholds should not be viewed as rendering our segmentation algorithm language-dependent.

pus frequency to determine whether the attachment of a morpheme to a root word is plausible or not.

Consider again the two words "candidate" and "candid". While "candidate" occurs 6380 times in our corpus, "candid" occurs only 119 times. This frequency disparity can be an important clue to determining that there is no morphological relation between "candidate" and "candid". Similar observation is also made by Yarowsky and Wicentowski (2000), who successfully employ relative frequency similarity or disparity to rank candidate VBD/VB pairs (e.g. "sang"/"sing") that are irregular in nature. Unlike Yarowsky and Wicentowski, however, our goal is to detect incorrect affix attachments and improve morphological analysis.

Our incorrect attachment detection algorithm, which exploits frequency disparity, is based on the following hypothesis: if a word w is formed by attaching an affix m to a root word r, then the corpus frequency of w is likely to be less than that of r (i.e. the frequency ratio of w to r is less than one). In other words, we hypothesize that the inflectional or derivational forms of a root word occur less frequently in a corpus than the root itself.

To illustrate this hypothesis, Table 1 shows some randomly chosen English words together with their *word-root frequency ratios* (WRFRs). The <word, root> pairs in the left side of the table are examples of correct attachments, whereas those in the right side are not. Note that only those words that represent correct attachments have a WRFR less than 1.

The question, then, is: to what extent does our hypothesis hold? To investigate this question, we generated examples of correct attachments by randomly selecting 400 words from our English vocabulary and then removing those that are root words, proper nouns, or compound words. We then manually segmented each of the remaining 378 words as Prefix+Root or Root+Suffix with the aid of the CELEX lexical database (Baayean et al., 1996). Somewhat surprisingly, we found that the WRFR is less than 1 in only 71.7% of these attachments. When the same experiment was repeated on 287 hand-segmented Bengali words, the hypothesis achieves a higher accuracy of 83.6%.

To better understand why our hypothesis does not work well for English, we measured its accuracy separately for the Root+Suffix words and the Prefix+Root words, and found that the hypothesis fails mostly on the suffixal attachments (see Table 2). Though surprising at first glance, the relatively poor accuracy on suffixal attachments can be attributed to the fact that many words in English (e.g. "amusement", "winner") appear more frequently in our corpus than their corresponding root forms (e.g. "amuse", "win"). For Bengali, our hypothesis fails mainly on verbs, whose inflected forms occur more often in our corpus than their roots. This violation of the hypothesis can be attributed to the grammatical rule that the main verb of a Bengali sentence has to be inflected according to the subject in order to maintain sentence order.

To improve the accuracy of our hypothesis on detecting correct attachments, we relax our initial hypothesis as follows: if an attachment is correct, then the corresponding WRFR is less than some predefined threshold t, where $t > 1$. However, we do not want t to be too large, since our algorithm may then determine many incorrect attachments as correct. In addition, since our hypothesis has a high accuracy for prefixal attachments than suffixal attachments, the threshold we employ for prefixes can be smaller than that for suffixes. Taking into account these considerations, we use a threshold of 10 for suffixes and 2 for prefixes for all the languages we consider in this paper.

Correct Parses			Incorrect Parses		
Word	**Root**	**WRFR**	**Word**	**Root**	**WRFR**
bear-able	bear	0.01	candid-ate	candid	53.6
attend-ance	attend	0.24	medic-al	medic	483.9
arrest-ing	arrest	0.06	prim-ary	prim	327.4
sub-group	group	0.0002	ac-cord	cord	24.0
re-cycle	cycle	0.028	ad-diction	diction	52.7
un-settle	settle	0.018	de-crease	crease	20.7

Table 1: Word-root frequency ratios

	Root+Suffix	Prefix+Root	Overall
# of words	344	34	378
WRFR < 1	70.1%	88.2%	71.7%

Table 2: Hypothesis validation for English

Now we can employ our hypothesis to detect incorrect attachments and improve root induction as follows. For each word, w, in the vocabulary, we check whether (1) w can be segmented as $r+x$ or $p+r$, where p and x are valid prefixes and suffixes respectively and r is another word in the vocabulary, and (2) the WRFR for w and r is less than our predefined thresholds (10 for suffixes and 2 for prefixes). If both conditions are satisfied, it means that w is divisible. Hence, we add w into the list of roots if at least one of the conditions is violated.

5 Suffix Level Similarity

Many of the incorrect suffixal attachments have a WRFR between 1 and 10, but the detection algorithm described in the previous section will determine all of them as correct attachments. Hence, in this section, we propose another technique, which we call *suffix level similarity*, to identify some of these incorrect attachments.

Suffix level similarity is motivated by the following observation: if a word *w* combines with a suffix *x*, then *w* should also combine with the suffixes that are "morphologically similar" to *x*. To exemplify, consider the suffix "ate" and the root word "candid". The words that combine with the suffix "ate" (e.g. "alien", "fabric", "origin") also combine with suffixes like "ated", "ation" and "s". Given this observation, the question of whether "candid" combines with the suffix "ate" then lies in whether or not "candid" combines with "ated", "s" and "ation". The fact that "candid" does not combine with many of the above suffixes provides suggestive evidence that "candidate" cannot be derived from "candid".

More specifically, to check whether a word *w* combines with a suffix *x* using suffix level simialrity, we (1) find the set of words W_x that can combine with *x*; (2) find the set of suffixes S_x that attach to all of the words in W_x under the constraint that S_x does not contain *x*; and (3) find the 10 suffixes in S_x that are most "similar" to *x*. The question, then, is how to define similarity. Intuitively, a good similarity metric should reflect, for instance, the fact that "ated" is a better suffix to consider in the attachment decision for "ate" than "s" (i.e. "ated" is more similar to "ate" than "s"), since "s" attaches to most nouns and verbs in English and hence should not be a distinguishing feature for incorrect attachment detection.

We employ a probabilistic measure (PM) that computes the similarity between suffixes *x* and *y* as the product of (1) the probability of a word combining with *y* given that it combines with *x* and (2) the probability of a word combining with *x* given that it combines with *y*. More specifically,

$$PM(x, y) = P(y \mid x) * P(x \mid y) = \frac{n}{n_1} * \frac{n}{n_2},$$

where n_1 is the number of distinct words that combine with *x*, n_2 is the number of distinct words that combine with *y*, and *n* is the number of distinct words that combine with both *x* and *y*.[4]

After getting the 10 suffixes that are most similar to *x*, we employ them as features and use the associated similarity values (we scale them linearly between 1 and 10) as the weights of these 10 features. The decision of whether a suffix *x* can attach to a word *w* depends on whether the following inequality is satisfied:

$$\sum_{1}^{10} f_i w_i > t,$$

where f_i is a boolean variable that has the value 1 if *w* combines with x_i, where x_i is one of the 10 suffixes that are most similar to *x*; w_i is the scaled similarity between *x* and x_i; and *t* is a predefined threshold that is greater than 0.

One potential problem with suffix level similarity is that it is an overly strict condition for those words that combine with only one or two suffixes in the vocabulary. For example, if the word "character" has just one variant in the vocabulary (e.g. "characters"), suffix level similarity will determine the attachment of "s" to "character" as incorrect, since the weighted sum in the above inequality will be 0. To address this sparseness problem, we rely on both relative corpus frequency and suffix level similarity to identify incorrect attachments. Specifically, if the WRFR of a <word, root> pair is between 1 and 10, we determine that an attachment to the root is incorrect if

$$-WRFR + \gamma * (\textit{suffix level similarity}) < 0,$$

where γ is set to 0.15.

Finally, since long words have a higher chance of getting segmented, we do not apply suffix level similarity to words whose length is greater than 10.

6 Inducing Orthographic Rules and Allomorphs

The biggest drawback of the system, described thus far, is its failure to segment words where the roots exhibit spelling changes during attachment to morphemes (e.g. "denial" = "deny"+"al"). The reasons are (1) the system does not have any knowledge of language-specific orthographic rules (e.g. in English, the character 'y' at the morpheme boundary is changed to 'i' when the root combines

[4] Note that this metric has the desirable property of returning a low similarity value for "s": while *n* is likely to be large, it will be offset by a large n_2.

with the suffix "al"), and (2) the vocabulary we employ for morpheme induction does not normally contain the *allomorphic variations* of the roots (e.g. "deni" is allomorphic variation of "deny"). To segment these words correctly, we need to generate the allomorphs and orthographic rules automatically given a set of induced roots and affixes.

Before giving the details of the generation method, we note that the induction of orthographic rules is a challenging problem, since different languages exhibit orthographic changes in different ways. For some languages (e.g. English) rules are mostly predictable, whereas for others (e.g. Finnish) rules are highly irregular. It is hard to obtain a generalized mapping function that aligns every <root, allomorph> pair, considering the fact that our system is unsupervised. An additional challenge is to ensure that the incorporation of these orthographic rules will not adversely affect system performance (i.e. they will not be applied to regular words and thus segment them incorrectly). Yarowsky and Wicentowski (2000) propose an interesting algorithm that employs four similarity measures to successfully identify the most probable root of a highly irregular word. Unlike them, however, our goal is to (1) check whether the learned rules can actually improve an unsupervised morphological system, not just to align <root, allomorph> pair, and (2) examine whether our system is extendable to different languages.

Taking into consideration the aforementioned challenges, our induction algorithm will (1) handle orthographic character changes that occur only at morpheme boundaries; (2) generate allomorphs for suffixal attachments only[5], assuming that roots exhibit the character changes during attachment, not suffixes; and (3) learn rules that aligns <root, allomorph> pairs of edit distance 1 (which may involve 1-character replacement, deletion or insertion). Despite these limitations, we will see that the incorporation of the induced rules improves segmentation accuracy significantly.

Let us first discuss how we learn a *replacement* rule, which identifies <allomorph, root> pairs where the last character of the root is replaced by another character. The steps are as follows:

(1) Inducing candidate allomorphs

If $\alpha A\beta$ is a word in the vocabulary (e.g. "denial", where α="den", A="i", and β="al"), β is an induced suffix, αB is an induced root (e.g. "deny", where B="y"), and the attachment of β to αB is correct according to relative corpus frequency (see Section 4), then we hypothesize that αA is an allomorph of αB. For each induced suffix, we use this hypothesis to generate the allomorphs and identify those that are generated from at least two suffixes as *candidate allomorphs*. We denote the list of <candidate allomorph, root, suffix> tuples by L.

(2) Learning orthographic rules

Every <candidate allomorph, root, suffix> tuple as learned above is associated with an orthographic rule. For example, from the words "denial", "deny" and suffix "al", we learn the rule "y:i / _ + al"[6]; from "social", "sock" and "al", we learn the rule "k:i / _ + al", which, however, is erroneous. So, we check whether each of the learned rules occurs frequently enough for all the <allomorph, root> pairs associated with a suffix, with the goal of filtering the low-frequency orthographic rules. Specifically, for each suffix β, we repeat the following steps:

(a) Counting the frequency of rules. Let L_β be the list of <candidate allomorph, root> pairs in L that are associated with the suffix β. For each pair p in L_β, we first check whether its candidate allomorph appears in any other <candidate allomorph, root> pairs in L_β. If not, we increment the frequency of the orthographic rule associated with p by 1. For example, the pair <"deni", "deny"> increases the frequency of the rule "y:i" by 1 on condition that "deni" does not appear in any other pairs.

(b) Filtering the rules. We first remove the infrequent rules, specifically those that are induced by less than 15% of the tuples in L_β. Then we check whether there exists two rules of the form A:B and A:C in the induced rule list. If so, then we have a morphologically undesirable situation where the character A changes to B and C under the same environment (i.e. β). To address this problem, we first calculate the strength of a rule as follows:

$$strength(A:B) = \frac{frequency(A:B)}{\sum_{@} frequency(A:@)}$$

We then retain only those rules whose frequency*strength is greater than some predefined threshold. We denote the list of rules that satisfy the above constraints by R_β.

(c) Identifying valid allomorphs. For each rule in R_β, we identify the associated <candidate allo-

[5] We only learn rules for suffixes of length greater than 1, since most suffixes of length 1 do not participate in orthographic changes.

[6] This is the Chomsky and Halle notation for representing orthographic rules. *a*:*b* / *c* _ *d* means *a* changes to *b* when the left context is *c* and the right context is *d*.

morph, root> pairs in L_β. We refer to the candidate allomorphs in each of those pairs as *valid allomorphs* and add them to the list of roots. We also remove from the original root list the words that can be segmented by the induced allomorphs and the associated rules (e.g. "denial").

(d) Identifying composite suffixes. For each rule in R_β, we also check whether it can identify composite suffixes where the first component suffix's last character is replaced during attachment to the second component suffix (e.g. "liness" = "ly"+"ness"). Specifically, if (1) $A{:}B$ / _ β is a rule in R_β, (2) $\alpha A\beta$ (say "liness"), β (say "ness") and αB (say "ly") are induced suffixes, and (3) $\alpha A\beta$ satisfies the requirements of a composite suffix (see Section 3.2), then we determine that $\alpha A\beta$ is a composite suffix composed of αB and β.

We employ the same procedure for learning *insertion* and *deletion* rules, except that strength is always set to 1 for these two types of rules. The threshold we set at step (b) is somewhat dependent on the vocabulary size, since the frequency count of each rule will naturally be larger when a larger vocabulary is used. Following our method for setting vocabulary-dependent thresholds (see Section 3.1), we set the threshold to 4 for English and 25 for Finnish, for instance.

Finally, we adapt our candidate allomorph detection method described above to induce allomorphs that are generated through orthographic changes of edit distance greater than 1. Specifically, if $\alpha\beta$ is a word in the induced root list (e.g. "stability"[7], where α="stabil" and β="ity"), β is an induced suffix, and the attachment of β to α is correct according to suffix level similarity, then we hypothesize that α ("stabil") is a candidate allomorph. For each induced suffix, we use this hypothesis to generate candidate allomorphs and consider as valid allomorphs only those that are generated from at least three different suffixes.[8]

7 Word Segmentation

After inducing the morphemes, we can use them to segment a word w in the test set. Specifically, we (1) *generate* all possible segmentations of w using only the induced affixes and roots, and (2) apply a sequence of tests to *remove* candidate segmentations until we are left with only one candidate, which we take to be the final segmentation of w.

Our first test involves removing any candidate segmentation $m_1m_2 \ldots m_n$ that violates any of the linguistic constraints below:

- At least one of $m_1, m_2, \ldots, m_n$ is a root.
- For $1 \le i < n$, if m_i is a prefix, then m_{i+1} must be a root or a prefix.
- For $1 < i \le n$, if m_i is a suffix, then m_{i-1} must be a root or a suffix.
- m_1 can't be a suffix and m_n can't be a prefix.

Next, we apply our second test, in which we retain only those candidate segmentations that have the smallest number of morphemes. For example, if "friendly" has two candidate segmentations "friend"+"ly" and "fri"+"end"+"ly", we will select the first one to be the segmentation of w.

If more than one candidate segmentation still exists, we score each remaining candidate using the heuristic below, selecting the highest-scoring candidate to be the final segmentation of w. Basically, we score each candidate segmentation by adding the *strength* of each morpheme in the segmentation, where (1) the strength of an affix is the number of distinct words in the vocabulary to which the affix attaches, multiplied by the length of the affix, and (2) the strength of a root is the number of distinct morphemes with which the root combines, again multiplied by the length of the root.

8 Evaluation

In this section, we will first evaluate our segmentation algorithm for English and Bengali, and then examine its performance on the PASCAL datasets.

8.1 Experimental Setup

Vocabulary creation. We extracted our English vocabulary from the Wall Street Journal corpus of the Penn Treebank and the BLLIP corpus, preprocessing the documents by first tokenizing them and then removing capitalized words, punctuations and numbers. In addition, we removed words of frequency 1 from BLLIP, because many of them are proper nouns and misspelled words. The final English vocabulary consists of approximately 60K distinct words. We applied the same pre-processing

[7] The correct segmentation of "stability" is "stable"+"ity". The "stabil"-"stable" allomorph-root pair is an example of an orthographic change of edit distance 2.

[8] This technique can also be used to induce out-of-vocabulary (OOV) roots. For example, the presence of "perplexity", "perplexed" and "perplexing" in a vocabulary allows us to induce the root "perplex". OOV root induction is particularly important for languages like Bengali, where verb roots mostly take the imperative form and hence are absent in a vocabulary created from a newspaper corpus, which normally comprises only the first and third person verb forms.

steps to five years of articles taken from the Bengali newspaper *Prothom Alo* to generate our Bengali vocabulary, which consists of 140K words.

Test set preparation. To create our English test set, we randomly chose 5000 words from our vocabulary that are at least 4-character long[9] and also appear in CELEX. Although 95% of the time we adopted the segmentation proposed by CELEX, in some cases the CELEX segmentations are erroneous (e.g. "rolling" and "lodging" remain unsegmented in CELEX). As a result, we cross-check with the online version of Merriam-Webster to make the necessary changes. To create the Bengali test set, we randomly chose 5000 words from our vocabulary and manually removed proper nouns and misspelled words from the set before giving it to two of our linguists for hand-segmentation. The final test set contains 4191 words.

Evaluation metrics. We use two standard metrics -- *exact accuracy* and *F-score* -- to evaluate the performance of our segmentation algorithm on the test sets. Exact accuracy is the percentage of the words whose proposed segmentation is identical to the correct segmentation. F-score is the harmonic mean of recall and precision, which are computed based on the placement of morpheme boundaries.[10]

8.2 Results for English and Bengali

The baseline systems. We use two publicly available and widely used unsupervised morphological learning systems -- Goldsmith's (2001) Linguistica[11] and Creutz and Lagus's (2005) Morphessor 1.0[12] -- as our baseline systems. The first two rows of Table 3 show the results of these systems for our test sets (with all the training parameters set to their default values). As we can see, Linguistica performs substantially better for English in terms of both exact accuracy and F-score, whereas Morphessor outperforms Linguistica for Bengali.

Our segmentation algorithm. Results of our segmentation algorithm are shown in rows 3-6 of Table 3. Specifically, row 3 shows the results of our basic segmentation system as described in Section 3. Rows 4-6 show the results where our three techniques (i.e. relative frequency, suffix level similarity and allomorph detection) are incorporated into the basic system one after the other. It is worth mentioning that (1) our basic algorithm already outperforms the baseline systems in terms of both exact accuracy and F-score; and (2) while each of our additions to the basic algorithm boosts system performance, relative corpus frequency and allomorph detection contribute to performance improvements particularly significantly. As we can see, the best segmentation performance is achieved when all of our three additions are applied to the basic algorithm.

	English				Bengali			
	A	P	R	F	A	P	R	F
Linguistica	68.9	84.8	75.7	80.0	36.3	58.2	63.3	60.6
Morphessor	64.9	69.6	85.3	76.6	56.5	89.7	67.4	76.9
Basic induction	68.1	79.4	82.8	81.1	57.7	79.6	81.2	80.4
Relative frequency	74.0	86.4	82.5	84.4	63.2	85.6	79.9	82.7
Suffix level similarity	74.9	88.6	82.3	85.3	66.1	89.7	78.8	83.9
Allomorph detection	**78.3**	88.3	86.4	**87.4**	**68.3**	89.3	81.3	**85.1**

Table 3: Results (reported in terms of exact accuracy (A), precision (P), recall (R) and F-score (F))

8.3 PASCAL Challenge Results

To get an idea of how our algorithm performs in comparison to the PASCAL participants, we conducted evaluations on the PASCAL datasets for English, Finnish and Turkish. Table 4 shows the F-scores of four segmentation algorithms for these three datasets: the best-performing PASCAL system (Winner), Morphessor, our system that uses the basic morpheme induction algorithm (Basic), and our system with all three extensions incorporated (Complete). Below we discuss these results.

English. There are 533 test cases in this dataset. Using the vocabulary created as described in Section 8.1, our Complete algorithm achieves an F-score of 79.4%, which outperforms the winner (Keshava and Pitler, 2006) by 2.6%. Although our basic morpheme induction algorithm is similar to that of Keshava and Pitler, a closer examination of the results reveals that F-score increases significantly with the incorporation of relative frequency and allomorph detection.

Finnish and Turkish. The real challenge in the PASCAL Challenge is the evaluation on Finnish

[9] Words of length less than 4 do not have any morphological segmentation in English. Hence, by imposing this length restriction on the words in our test set, we effectively make the evaluation more challenging. This is also the reason for our using words that are at least 3-character long in the Bengali test set.
[10] See http://www.cis.hut.fi/morphochallenge2005/evaluation.shtml for details.
[11] http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/
[12] http://www.cis.hut.fi/projects/morpho/

and Turkish due to their morphological richness. We use the 400K and 300K most frequent words from the Finnish and Turkish datasets provided by the organizers as our vocabulary. When tested on the gold standard of 661 Finnish and 775 Turkish words, our Complete system achieves F-scores of 65.2% and 66.2%, which are better than the winner's scores (Bernhard (2006)). In addition, Complete outperforms Basic by 3-6% in F-score; these results suggest that the new techniques proposed in this paper (especially allomorph detection) are also very effective for Finnish and Turkish.

	English	**Finnish**	**Turkish**
Winner	76.8	64.7	65.3
Morphessor	66.2	66.4	70.1
Basic	75.8	59.2	63.4
Complete	79.4	65.2	66.2

Table 4: F-scores for the PASCAL gold standards

As mentioned in the introduction, none of the participating PASCAL systems offers robust performance across different languages. For instance, Keshava and Pitler's algorithm, the winner for English, has F-scores of only 47% and 54% for Finnish and Turkish respectively, whereas Bernhard's algorithm, the winner for Finnish and Turkish, achieves an F-score of only 66% for English. On the other hand, our algorithm outperforms the winners for *all* the languages in the competition, demonstrating its robustness across languages.

Finally, although Morphessor achieves better results for Turkish and Finnish than our Complete system, it performs poorly for English, having an F-score of only 66.2%. On the other hand, our results for Finnish and Turkish are not significantly poorer than those of Morphessor.

9 Conclusions

We have presented an unsupervised word segmentation algorithm that offers robust performance across languages with different levels of morphological complexity. Our algorithm not only outperforms Linguistica and Morphessor for English and Bengali, but also compares favorably to the best-performing PASCAL morphological parsers when evaluated against all three target languages -- English, Turkish, and Finnish -- in the Challenge. Experimental results indicate that the use of relative corpus frequency for incorrect attachment detection and the induction of orthographic rules and allomorphs have contributed to the performance of our algorithm particularly significantly.

References

R. H. Baayen, R. Piepenbrock and L. Gulikers. 1996. The CELEX2 lexical database (CD-ROM), LDC, Univ of Pennsylvania, Philadephia, PA.

D. Bernhard. 2006. Unsupervised morphological segmentation based on segment predictability and word segment alignment. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

M. R. Brent, S. K. Murthy and A. Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the Fifth International Workshop on AI and Statistics*.

M. Creutz. 2003. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the ACL,* pages 280-287.

M. Creutz and K. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. In *Computer and Information Science, Report A81*, Helsinki University of Technology.

S. Dasgupta and V. Ng. 2007. Unsupervised word segmentation for Bangla. In *Proceedings of ICON*, pages 15-24.

H. DéJean. 1998. Morphemes as necessary concepts for structures discovery from untagged corpora. In *Workshop on Paradigms and Grounding in Natural Language Learning*, pages 295-299.

D. Freitag. 2005. Morphology induction from term clusters. In *Proceedings of CoNLL*, pages 128-135.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. In *Computational Linguistics* 27(2), pages 153-198.

Z. Harris. 1955. From phoneme to morpheme. In *Language*, 31(2): 190-222.

S. Keshava and E. Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

K. Koskenniemi. 1983. Two-level morphology: a general computational model for word-form recognition and production. *Publication No. 11. Helsinki: University of Helsinki Department of General Linguistics.*

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the NAACL*, pages 183-191.

M. G. Snover and M. R. Brent. 2001. A Bayesian model for morpheme and paradigm identification. In *Proceedings of the ACL*, pages 482-490.

D. Yarowsky and R. Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the ACL*, pages 207-216.

Probabilistic Generation of Weather Forecast Texts

Anja Belz
Natural Language Technology Group
University of Brighton, UK
a.s.belz@brighton.ac.uk

Abstract

This paper reports experiments in which *p*CRU — a generation framework that combines probabilistic generation methodology with a comprehensive model of the generation space — is used to semi-automatically create several versions of a weather forecast text generator. The generators are evaluated in terms of output quality, development time and computational efficiency against (i) human forecasters, (ii) a traditional handcrafted pipelined NLG system, and (iii) a HALOGEN-style statistical generator. The most striking result is that despite acquiring all decision-making abilities automatically, the best *p*CRU generators receive higher scores from human judges than forecasts written by experts.

1 Introduction and background

Over the last decade, there has been a lot of interest in statistical techniques among researchers in natural language generation (NLG), a field that was largely unaffected by the statistical revolution in NLP that started in the 1980s. Since Langkilde and Knight's influential work on statistical surface realisation (Knight and Langkilde, 1998), a number of statistical and corpus-based methods have been reported. However, this interest does not appear to have translated into practice: of the 30 implemented systems and modules with development starting in or after 2000 that are listed on a key NLG website[1], only five have any statistical component at all (another six involve techniques that are in some way corpus-based). The likely reasons for this lack of take-up are that (i) many existing statistical NLG techniques are inherently expensive, requiring the set of alternatives to be generated in full before the statistical model is applied to select the most likely; and (ii) statistical NLG techniques have not been shown to produce outputs of high enough quality.

There has also been a rethinking of the traditional modular NLG architecture (Reiter, 1994). Some research has moved towards a more comprehensive view, e.g. construing the generation task as a single constraint satisfaction problem. Precursors to current approaches were Hovy's PAULINE which kept track of the satisfaction status of global 'rhetorical goals' (Hovy, 1988), and Power et al.'s ICONOCLAST which allowed users to fine-tune different combinations of global constraints (Power, 2000). In recent comprehensive approaches, the focus is on automatic adaptability, e.g. automatically determining degrees of constraint violability on the basis of corpus frequencies. Examples include Langkilde's (2005) general approach to generation and parsing based on constraint optimisation, and Marciniak and Strube's (2005) integrated, globally optimisable network of classifiers and constraints.

Both probabilistic and recent comprehensive trends have developed at least in part to address two interrelated issues in NLG: the considerable amount

[1]Bateman and Zock's list of NLG systems, `http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/`, 20/01/2006.

of time and expense involved in building new systems, and the almost complete lack in the field of reusable systems and modules. Both trends have the potential to improve on development time and reusability, but have drawbacks.

Existing statistical NLG (i) uses corpus statistics to inform heuristic decisions in what is otherwise symbolic generation (Varges and Mellish, 2001; White, 2004; Paiva and Evans, 2005); (ii) applies *n*-gram models to select the overall most likely realisation after generation (HALOGEN family); or (iii) reuses an existing parsing grammar or treebank for surface realisation (Velldal et al., 2004; Cahill and van Genabith, 2006). N-gram models are not linguistically informed, (i) and (iii) come with a substantial manual overhead, and (ii) overgenerates vastly and has a high computational cost (see also Section 3).

Existing comprehensive approaches tend to incur a manual overhead (finetuning in ICONOCLAST, corpus annotation in Langkilde and Marciniak & Strube). Handling violability of soft constraints is problematic, and converting corpus-derived probabilities into costs associated with constraints (Langkilde, Marciniak & Strube) turns straightforward statistics into an *ad hoc* search heuristic. Older approaches are not globally optimisable (PAULINE) or involve exhaustive search (ICONOCLAST).

The *p*CRU language generation framework combines a probabilistic generation methodology with a comprehensive model of the generation space, where probabilistic choice informs generation as it goes along, instead of after all alternatives have been generated. *p*CRU uses existing techniques (Belz, 2005), but extends these substantially. This paper describes the *p*CRU framework and reports experiments designed to rigorously test *p*CRU in practice and to determine whether improvements in development time and reusability can be achieved without sacrificing quality of outputs.

2 *p*CRU language generation

*p*CRU (Belz, 2006) is a probabilistic language generation framework that was developed with the aim of providing the formal underpinnings for creating NLG systems that are driven by comprehensive probabilistic models of the entire generation space (including deep generation). NLG systems tend to be composed of generation rules that apply transformations to representations (performing different tasks in different modules). The basic idea in *p*CRU is that as long as the generation rules are all of the form $relation(arg_1, ...arg_n) \rightarrow relation_1(arg_1, ...arg_p)$ $... \ relation_m(arg_1, ...arg_q)$, $m \geq 1, n, p, q \geq 0$, then the set of all generation rules can be seen as defining a context-free language and a single probabilistic model can be estimated from raw or annotated text to guide generation processes.

*p*CRU uses straightforward context-free technology in combination with underspecification techniques, to encode a **base generator** as a set of expansion rules G composed of n-ary relations with variable and constant arguments (Section 2.1). In non-probabilistic mode, the output is the set of fully expanded (fully specified) forms that can be derived from the input. The *p*CRU (*p*robabilistic CRU) **decision-maker** is created by estimating a probability distribution over the base generator from an unannotated corpus of example texts. This distribution is used in one of several ways to drive generation processes, maximising the likelihood either of individual expansions or of entire generation processes (Section 2.2).

2.1 Specifying the range of alternatives

Using context-free representational underspecification, or CRU, (Belz, 2004), the generation space is encoded as (i) a set G of expansion rules composed of n-ary relations $relation(arg_1, ...arg_n)$ where the arg_i are constants or variables over constants; and (ii) argument and relation type hierarchies. Any sentential form licensed by G can be the input to the generation process which expands it under unifying variable substitution until no further expansion is possible. The output (in non-probabilistic mode) is the set of fully expanded forms (i.e. consisting only of terminals) that can be derived from the input.

The rules in G define the steps in which inputs can be incrementally specified from, say, content to semantic, syntactic and finally surface representations. G therefore defines specificity relations between all sentential forms, i.e. defines which representation is underspecified with respect to which other representations. The generation process is construed explicitly as the task of incrementally specifying one or more word strings.

Within the limits of context-freeness and atomicity of feature values, CRU is neutral with respect to actual linguistic knowledge representation formalisms used to encode generation spaces. The main motivation for a context-free formalism is the advantage of low computational cost, while the inclusion of arguments on (non)terminals permits keeping track of contextual features.

2.2 Selection among alternatives

The *p*CRU decision-making component is created by estimating a probability distribution over the set of expansion rules that encodes the generation space (the base generator), as follows:

1 *Convert corpus into multi-treebank:* determine for each sentence all (left-most) derivation trees licensed by the base generator's CRU rules, using maximal partial derivations if there is no complete derivation tree; annotate the (sub)strings in the sentence with the derivation trees, resulting in a set of *generation trees* for the sentence.

2 *Train base generator:* Obtain frequency counts for each individual generation rule from the multi-treebank, adding $1/n$ to the count for every rule, where n is the number of alternative derivation trees; convert counts into probability distributions over alternative rules, using add-1 smoothing and standard maximum likelihood estimation.

The resulting probability distribution is used in one of the following three ways to control generation. Of these, only the first requires the generation forest to be created in full, whereas both greedy modes prune the generation space to a single path:

1 *Viterbi generation:* do a Viterbi search of the generation forest for a given input, which maximises the joint likelihood of all decisions taken in the generation process. This selects the most likely generation process, but is considerably more expensive than the greedy modes.

2 *Greedy generation:* make the single most likely decision at each choice point (rule expansion) in a generation process. This is not guaranteed to result in the most likely generation process, but the computational cost is very low.

3 *Greedy roulette-wheel generation:* use a non-uniform random distribution proportional to the likelihoods of alternatives. E.g. if there are two alternative decisions D_1 and D_2, with the model giving $p(D_1) = 0.8$ and $p(D_2) = 0.2$, then the proportion of times the generator decides D_1 approaches 80% and D_2 20% in the limit.

2.3 The *p*CRU-1.0 generation package

The technology described in the two preceding sections has been implemented in the *p*CRU-1.0 software package. The user defines a generation space by creating a base generator composed of:

1. the set N of underspecified n-ary relations
2. the set W of fully specified n-ary relations
3. a set R of context-free generation rules $n \rightarrow \alpha$, $n \in N$, $\alpha \in (W \cup N)^*$
4. a typed feature hierarchy defining argument types and values

This base generator is then trained (as described above) on raw text corpora to provide a probability distribution over generation rules. Optionally, an *n*-gram language model can also be created from the same corpus. The generator is then run in one of the three modes above or one of the following:

1. *Random*: ignoring *p*CRU probabilities, randomly select generation rules.
2. *N-gram*: ignoring *p*CRU probabilities, generate set of alternatives and select the most likely according to the *n*-gram language model.

The random mode serves as a baseline for generation quality: a trained generator must be able to do better, otherwise all the work is done by the base generator (and none by the probabilities). The *n*-gram mode works exactly like HALOGEN-style generation: the generator generates all realisations that the rules allow and then picks one based on the *n*-gram model. This is a point of comparison with existing statistical NLG techniques and also serves as a baseline in terms of computational expense: a generator using *p*CRU probabilities should be able to produce realisations faster.

3 Building and evaluating *p*CRU wind forecast text generators

The automatic generation of weather forecasts is one of the success stories of NLP. The restrictiveness of the sublanguage has made the domain of

```
Oil1/Oil2/Oil3_FIELDS
05-10-00

05/06 SSW 18  22  27   3.0  4.8 SSW  2.59
05/09 S   16  20  25   2.7  4.3 SSW  2.39
05/12 S   14  17  21   2.5  4.0 SSW  2.29
05/15 S   14  17  21   2.3  3.7 SSW  2.28
...


FORECAST FOR:-
Oil1/Oil2/Oil3 FIELDS
...
2.FORECAST 06-24 GMT, THURSDAY, 05-Oct   2000
=====WARNINGS:   RISK THUNDERSTORM.  =======
WIND(KTS)   CONFIDENCE: HIGH
  10M:      SSW 16-20 GRADUALLY BACKING SSE
            THEN FALLING VARIABLE 04-08 BY
            LATE EVENING
...
```

Figure 1: Meteorological data file and wind forecast for 05-10-2000, a.m. (oil fields anonymised).

weather forecasting particularly attractive to NLG researchers, and a number of weather forecast generation systems have been created.

A recent example of weather forecast text generation is the SUMTIME project (Reiter et al., 2005) which developed a commercially used NLG system that generates marine weather forecasts for offshore oil rigs from numerical forecast data produced by weather simulation programs. The SUMTIME corpus is used in the experiments below.

3.1 Data

Each instance in the SUMTIME corpus consists of three numerical data files (the outputs of weather simulators) and the forecast file written by the forecaster on the basis of the data (Figure 1 shows an example). The experiments below focused on a.m. forecasts of wind characteristics. Content determination (deciding which meteorological data to include in a forecast) was carried out off-line.

The corpus consists of 2,123 instances (22,985 words) of which half are a.m. forecasts. This may not seem much, but considering the small number of vocabulary items and syntactic structures, the corpus provides extremely good coverage (an initial impression confirmed by the small differences between training and testing data results below).

3.2 The base generator

The base generator[2] was written semi-automatically in two steps. First, a simple chunker was run over the corpus to split wind statements into wind direction, wind speed, gust speed, gust statements, time expressions, verb phrases, pre-modifiers, and post-modifiers. Preterminal generation rules were automatically created from the resulting chunks. Then, higher-level rules which combine chunks into larger components, taking care of text structuring, aggregation and elision, were manually authored. The top-level generation rules interpret wind statements as sequences of independent units of information, ensuring a linear increase in complexity with increasing input length. Inputs encode meteorological data (as shown in Table 1), and were pre-processed to determine certain types of information, including whether a change in wind direction was clockwise or anti-clockwise, and whether change in wind speed was an increase or a decrease. The final generator takes as inputs number vectors of length 7 to 60, and generates up to 1.6×10^{31} alternative realisations for an input.

The job of the base generator is to describe the textual variety found in the corpus. It makes no decisions about when to prefer one variant over another.

3.3 Training

The corpus was divided at random into 90% training data and 10% testing data. The training set was multi-treebanked with the base generator and the multi-treebank then used to create the probability distribution for the base generator (as described in Section 2.2). A back-off 2-gram model with Good-Turing discounting and no lexical classes was also created from the training set, using the SRILM toolkit, (Stolcke, 2002). *p*CRU-1.0 was then run in all five modes to generate forecasts for the inputs in both training and test sets.

This procedure was repeated five times for hold-out cross-validation. The small amount of variation across the five repeats, and the small differences between results for training and test sets (Table 2) indicated that five repeats were sufficient.

3.4 Evaluation

3.4.1 Evaluation methods

The two automatic metrics used in the evaluations, NIST and BLEU have been shown to correlate highly with expert judgments (Pearson correlation coefficients 0.82 and 0.79 respectively) in this domain (Belz and Reiter, 2006).

[2]For a fragment of the rule set, see Belz (2006).

Input	`[[1,SSW,16,20,-,-,0600],[2,SSE,-,-,-,-,NOTIME],[3,VAR,04,08,-,-,2400]]`
Corpus	`SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 4-8 BY LATE EVENING`
Reference 1	`SSW'LY 16-20 GRADUALLY BACKING SSE'LY THEN DECREASING VARIABLE 4-8 BY LATE EVENING`
Reference 2	`SSW 16-20 GRADUALLY BACKING SSE BY 1800 THEN FALLING VARIABLE 4-8 BY LATE EVENING`
SUMTIME-Hyb.	`SSW 16-20 GRADUALLY BACKING SSE THEN BECOMING VARIABLE 10 OR LESS BY MIDNIGHT`
*p*CRU-greedy	`SSW 16-20 BACKING SSE FOR A TIME THEN FALLING VARIABLE 4-8 BY LATE EVENING`
*p*CRU-roulette	`SSW 16-20 GRADUALLY BACKING SSE AND VARIABLE 4-8`
*p*CRU-viterbi	`SSW 16-20 BACKING SSE VARIABLE 4-8 LATER`
*p*CRU-2gram	`SSW 16-20 BACKING SSE VARIABLE 4-8 LATER`
*p*CRU-random	`SSW 16-20 AT FIRST FROM MIDDAY BECOMING SSE DURING THE AFTERNOON THEN VARIABLE 4-8`

Table 1: Forecast texts (for 05-10-2000) generated by each of the *p*CRU generators, the SUMTIME-Hybrid system and three experts. The corresponding input to the generators is shown in the first row.

BLEU (Papineni et al., 2002) is a precision metric that assesses the quality of a translation in terms of the proportion of its word *n*-grams ($n \leq 4$ has become standard) that it shares with several reference translations. BLEU also incorporates a 'brevity penalty' to counteract scores increasing as length decreases. BLEU scores range from 0 to 1.

The NIST metric (Doddington, 2002) is an adaptation of BLEU, but where BLEU gives equal weight to all *n*-grams, NIST gives more weight to less frequent (hence more informative) *n*-grams. There is evidence that NIST correlates better with human judgments than BLEU (Doddington, 2002; Belz and Reiter, 2006).

The results below include human scores from two separate experiments. The first was an experiment with 9 subjects experienced in reading marine forecasts (Belz and Reiter, 2006), the second is a new experiment with 14 similarly experienced subjects[3]. The main differences were that in Experiment 1, subjects rated on a scale from 0 to 5 and were asked for overall quality scores, whereas in Experiment 2, subjects rated on a 1–7 scale and were asked for language quality scores.

In comparing different *p*CRU modes, NIST and BLEU scores were computed against the test set part of the corpus which contains texts by five different authors. In the two human experiments, NIST and BLEU scores were computed against sets of multiple reference texts (2 for each date in Experiment 1, and 3 in Experiment 2) written by forecasters who had not contributed to the corpus. One-way ANOVAs with post-hoc Tukey HSD tests were used to analyse variance and statistical significance of all results.

Table 1 shows forecast texts generated by each of the systems included in the evaluations reported below, together with the corresponding input and three texts created by humans for the same data.

[3]Belz and Reiter, in preparation.

		NIST-5	BLEU-4
T	*p*CRU-greedy	8.208 (0.033)	0.647 (0.002)
R	*p*CRU-roulette	7.035 (0.138)	0.496 (0.010)
A	*p*CRU-2gram	6.734 (0.086)	0.523 (0.008)
I	*p*CRU-viterbi	6.643 (0.023)	0.524 (0.002)
N	*p*CRU-random	4.799 (0.036)	0.296 (0.002)
	*p*CRU-greedy	6.927 (0.131)	0.636 (0.016)
T	*p*CRU-roulette	6.193 (0.121)	0.496 (0.022)
E	*p*CRU-2gram	5.663 (0.185)	0.514 (0.019)
S	*p*CRU-viterbi	5.650 (0.161)	0.519 (0.021)
T	*p*CRU-random	4.535 (0.078)	0.313 (0.005)

Table 2: NIST-5 and BLEU-4 scores for training and test sets (average variation from the mean).

3.4.2 Comparing different generation modes

Table 2 shows results for the five different *p*CRU generation modes, for training sets (top) and test sets (bottom), in terms of NIST-5 and BLEU-4 scores averaged over the five runs of the hold-out validation, with average mean deviation figures across the runs shown in brackets.

The Tukey Test produced the following results for the differences between means in Table 2. For the training set, results are the same for NIST and BLEU scores: all differences are significant at $P < 0.01$, except for the differences in scores for *p*CRU-2gram and *p*CRU-viterbi. For the test set and NIST, again all differences are significant at $P < 0.01$, except for *p*CRU-2gram vs. *p*CRU-viterbi. For the test set and BLEU, three differences are non-significant: *p*CRU-2gram vs. *p*CRU-viterbi, *p*CRU-2gram vs. *p*CRU-

	Experiment 1	Experiment 2
SUMTIME-Hyb.	3.82 (1)	4.61 (2)
*p*CRU-greedy	3.59 (2)	4.79 (1)
*p*CRU-roulette	3.22 (3)	4.54 (3)

Table 3: Scores for handcrafted system and two best *p*CRU-systems from two human experiments.

roulette, and *p*CRU-viterbi vs. *p*CRU-roulette.

NIST-5 depends on test set size, and is necessarily lower for the (smaller) test set, but the BLEU-4 scores indicate that performance was slightly worse on test sets. The deviation figures show that variation was also higher on the test sets.

The clearest result is that *p*CRU-greedy is ranked highest, and *p*CRU-random lowest, by considerable margins. *p*CRU-roulette is ranked second by NIST-5 and fourth by BLEU-4. *p*CRU-2gram and *p*CRU-viterbi are virtually indistinguishable.

Experts in both human experiments agreed with the NIST-5 rankings of the modes exactly.

3.4.3 Text quality against handcrafted system

The *p*CRU modes were also evaluated against the SUMTIME-Hybrid system (running in 'hybrid' mode, taking inputs as in Table 1). Table 3 shows averaged evaluation scores by subjects in the two independent experiments described above. There were altogether 6 and 7 systems evaluated in these experiments, respectively, and the differences between the scores shown here were not significant when subjected to the Tukey Test, meaning that both experiments failed to show that experts can tell the difference in the language quality of the texts generated by the handcrafted SUMTIME-Hybrid system and the two best *p*CRU-greedy systems.

3.4.4 Text quality against human forecasters

In the first experiment, the human evaluators gave an average score of 3.59 to *p*CRU-greedy, 3.22 to the corpus texts, and 3.03 to another (human) forecaster. In Experiment 2, the average human scores were 4.79 for *p*CRU-greedy, and 4.50 for the corpus texts. Although in each experiment separately, statistical significance could not be shown for the differences between these means, in combination the scores provide evidence that the evaluators thought *p*CRU-greedy better than the human-written texts.

3.4.5 Computing time

The following table shows average number of seconds taken to generate one forecast, averaged over the five cross-validation runs (mean variation figures across the runs in brackets):

	Training sets	Test sets
*p*CRU-greedy:	1.65s $(= 0.02)$	1.58s (< 0.04)
*p*CRU-roulette:	1.61s (< 0.02)	1.58s (< 0.05)
*p*CRU-viterbi:	1.74s (< 0.02)	1.70s $(= 0.04)$
*p*CRU-2gram:	2.83s (< 0.02)	2.78s (< 0.09)

Forecasts for the test sets were generated somewhat faster than for the training sets in all modes. Variation was greater for test sets. Differences between *p*CRU-greedy and *p*CRU-roulette are very small, but *p*CRU-viterbi took $1/10$ of a second longer, and *p*CRU-2gram took more than 1 second longer to generate the average forecast[4].

3.4.6 Brevity bias

N-gram models have a built-in bias in favour of shorter strings, because they calculate the likelihood of a string of words as the joint probability of the words, or, more precisely, as the product of the probabilities of each word given the $n - 1$ preceding words. The likelihood of any string will therefore generally be lower than that of any of its substrings.

Using a smaller data set for which all systems had outputs, the average number of words in the forecasts generated by the different systems was:

*p*CRU-random:	19.43
SUMTIME-Hybrid:	12.39
*p*CRU-greedy:	11.51
Corpus:	*11.28*
*p*CRU-roulette:	10.48
*p*CRU-2gram:	7.66
*p*CRU-viterbi:	7.54

*p*CRU-random has no preference for shorter strings, its average string length is almost twice that of the other *p*CRU-generators. The 2-gram generator prefers shorter strings, while the Viterbi generator prefers shorter generation processes, and these preferences result in the shortest texts. The poor evaluation results above for the n-gram and Viterbi generators indicate that this brevity bias can be harm-

[4]The Viterbi and the 2-gram generator were implemented identically, except for the *n*-gram model look-up.

ful in NLG. The remaining generators achieve good matches to the average forecast length in the corpus.

3.4.7 Development time

The most time-consuming part of NLG system development is not encoding the range of alternatives, but the decision-making capabilities that enable selection among them. In SUMTIME (Section 3), these were the result of corpus analysis and consultation with writers and readers of marine forecasts. In the *p*CRU wind forecast generators, the decision-making capabilities are acquired automatically, no expert knowledge or corpus annotation is used.

The SUMTIME team estimate[5] that very approximately 12 person months went directly into developing the SUMTIME microplanner and realiser (the components functionally analogous to the *p*CRU-generators), and 24 on generic activities such as expert consultation, which also benefited the microplanner/realiser. The *p*CRU wind forecasters were built in less than a month, including familiarisation with the corpus, building the chunker and creating the generation rules themselves. However, the SUMTIME system also generates wave forecasts and appropriate layout and canned text. A generous estimate is that it would take another two person months to equip the *p*CRU forecaster with these capabilities.

This is not to say that the two research efforts resulted in exactly the same thing. It is clear that forecast readers prefer the SUMTIME system, but the point is that it did come with a substantial price tag attached. The *p*CRU approach allows control over the trade-off between cost and quality.

4 Discussion

The main contributions of the research described in this paper are: (i) a generation methodology that improves substantially on development time and reusability compared to traditional hand-crafted systems; (ii) techniques for training linguistically informed decision-making components for probabilistic NLG from raw corpora; and (iii) results that show that probabilistic NLG can produce high-quality text. Results also show that (i) a preference for shorter realisations can be harmful in NLG; and that (ii) linguistically literate, probabilistic NLG can outperform HALOGEN-style shallow statistical methods, in terms of quality and efficiency.

An interesting question concerns the contribution of the manually built component (the base generator) to the quality of the outputs. The random mode serves as an absolute baseline in this respect: it indicates how well a particular base generator performs on its own. However, different base generators have different effects on the generation modes. The base generator that was used in previous experiments (Belz, 2005) encoded a less structured generation space and the set of concepts it used were less fine-grained (e.g. it did not distinguish between an increase and a decrease in wind speed, considering both simply a change), and therefore it lacked some information necessary for deriving conditional probabilities for lexical choice (e.g. *freshening* vs. *easing*). As predicted (Belz, 2005, p. 21), improvements to the base generator made little difference to the results for *p*CRU-2gram (up from BLEU 0.45 to 0.5), but greatly improved the performance of the greedy mode (up from 0.43 to 0.64).

A basic question for statistical NLG is whether surface string likelihoods are enough to resolve remaining non-determinism in generators, or whether likelihoods at the more abstract level of generation rules are needed. The former always prefers the most frequent variant regardless of context, whereas in the latter probabilities can attach to linguistic objects and be conditioned on contextual features (e.g. one useful feature in the forecast text generators encoded whether a rule was being applied at the beginning of a text). The results reported in this paper provide evidence that probabilistic generation can be more powerful than n-gram based post-selection.

5 Conclusions

The *p*CRU approach to generation makes it possible to combine the potential accuracy and subtlety of symbolic generation rules with detailed linguistic features on the one hand, and the robustness and handle on nondeterminism provided by probabilities associated with these rules, on the other. The evaluation results for the *p*CRU generators show that outputs of high quality can be produced with this approach, that it can speed up development and improve reusability of systems, and that in some modes

[5]Personal communication with E. Reiter and S. Sripada.

it is more efficient and less brevity-biased than existing HALOGEN-style *n*-gram techniques.

The current situation in NLG recalls NLU in the late 1980s, when symbolic and statistical NLP were separate research paradigms, a situation memorably caricatured by Gazdar (1996), before rapidly moving towards a paradigm merger in the early 1990s. A similar development is currently underway in MT where — after several years of statistical MT dominating the field — researchers are now beginning to bring linguistic knowledge into statistical techniques (Charniak et al., 2003; Huang et al., 2006), and this trend looks set to continue. The lesson from NLU and MT appears to be that higher quality results when the symbolic and statistical paradigms join forces. The research reported in this paper is intended to be a first step in this direction for NLG.

Acknowledgments

This research was in part supported under UK EPSRC Grant GR/S24480/01. Many thanks to the anonymous reviewers for very helpful comments.

References

A. Belz and E. Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proc. EACL'06*, pages 313–320.

A. Belz. 2004. Context-free representational underspecification for NLG. Technical Report ITRI-04-08, University of Brighton.

A. Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proc. of ENLG'05*, pages 15–23.

A. Belz. 2006. *p*CRU: Probabilistic generation using representational underspecification. Technical Report NLTG-06-01, University of Brighton.

A. Cahill and J. van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proc. ACL'06*, pages 1033–44.

E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based language models for machine translation. In *Proc. MT Summit IX*.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the ARPA Workshop on Human Language Technology*.

G. Gazdar. 1996. Paradigm merger in NLP. In Robin Milner and Ian Wand, editors, *Computing Tomorrow: Future Research Directions in Computer Science*, pages 88–109. Cambridge University Press.

E. Hovy. 1988. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum.

L. Huang, K. Knight, and A. Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*, pages 66–73.

K. Knight and I. Langkilde. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of COLING-ACL'98*, pages 704–710.

I. Langkilde. 2005. An exploratory application of constraint optimization in Mozart to probabilistic natural language processing. In *Proceedings of CSLP'05*, volume 3438 of *LNAI*. Springer-Verlag.

T. Marciniak and M. Strube. 2005. Using an annotated corpus as a knowledge source for language generation. In *Proceedings of UCNLG'05*, pages 19–24.

D. S. Paiva and R. Evans. 2005. Empirically-based control of natural language generation. In *Proceedings ACL'05*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. ACL '02*, pages 311–318.

R. Power. 2000. Planning texts by constraint satisfaction. In *Proceedings of COLING'00*.

E. Reiter, S. Sripada, J. Hunter, and J. Yu. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

E. Reiter. 1994. Has a consensus NL generation architecture appeared and is it psycholinguistically plausible? In *Proceedings of INLG'94*, pages 163–170.

A. Stolcke. 2002. SRILM: An extensible language modeling toolkit. In *Proceedings of ICSLP'02*, pages 901–904,.

S. Varges and C. Mellish. 2001. Instance-based NLG. In *Proc. of NAACL'01*, pages 1–8.

E. Velldal, S. Oepen, and D. Flickinger. 2004. Paraphrasing treebanks for stochastic realization ranking. In *Proc. of TLT'04*.

M. White. 2004. Reining in CCG chart realization. In *Proceedings INLG'04*, volume 3123 of *LNAI*, pages 182–191. Springer.

Generation by Inverting a Semantic Parser That Uses Statistical Machine Translation

Yuk Wah Wong and Raymond J. Mooney
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233, USA
{ywwong,mooney}@cs.utexas.edu

Abstract

This paper explores the use of statistical machine translation (SMT) methods for tactical natural language generation. We present results on using phrase-based SMT for learning to map meaning representations to natural language. Improved results are obtained by inverting a semantic parser that uses SMT methods to map sentences into meaning representations. Finally, we show that hybridizing these two approaches results in still more accurate generation systems. Automatic and human evaluation of generated sentences are presented across two domains and four languages.

1 Introduction

This paper explores the use of statistical machine translation (SMT) methods in natural language generation (NLG), specifically the task of mapping statements in a formal *meaning representation language* (MRL) into a natural language (NL), i.e. tactical generation. Given a corpus of NL sentences each paired with a formal *meaning representation* (MR), it is easy to use SMT to construct a tactical generator, i.e. a statistical model that translates MRL to NL. However, there has been little, if any, research on exploiting recent SMT methods for NLG.

In this paper we present results on using a recent phrase-based SMT system, PHARAOH (Koehn et al., 2003), for NLG.[1] Although moderately effective, the inability of PHARAOH to exploit the formal structure and grammar of the MRL limits its accuracy. Unlike natural languages, MRLs typically have a simple, formal syntax to support effective automated processing and inference. This MRL structure can also be used to improve language generation.

Tactical generation can also be seen as the inverse of *semantic parsing*, the task of mapping NL sentences to MRs. In this paper, we show how to "invert" a recent SMT-based semantic parser, WASP (Wong and Mooney, 2006), in order to produce a more effective generation system. WASP exploits the formal syntax of the MRL by learning a translator (based on a statistical synchronous context-free grammar) that maps an NL sentence to a linearized parse-tree of its MR rather than to a flat MR string. In addition to exploiting the formal MRL grammar, our approach also allows the same learned grammar to be used for both parsing and generation, an elegant property that has been widely advocated (Kay, 1975; Jacobs, 1985; Shieber, 1988). We present experimental results in two domains previously used to test WASP's semantic parsing ability: mapping NL queries to a formal database query language, and mapping NL soccer coaching instructions to a formal robot command language. WASP^{-1} is shown to produce a more accurate NL generator than PHARAOH.

We also show how the idea of generating from linearized parse-trees rather than flat MRs, used effectively in WASP^{-1}, can also be exploited in PHARAOH. A version of PHARAOH that exploits this approach is experimentally shown to produce more accurate generators that are more competitive with WASP^{-1}'s. Finally, we also show how

[1]We also tried IBM Model 4/REWRITE (Germann, 2003), a word-based SMT system, but it gave much worse results.

```
((bowner our {4})
 (do our {6} (pos (left (half our)))))
```
If our player 4 has the ball, then our player 6 should stay in the left side of our half.

(a) CLANG

```
answer(state(traverse_1(riverid('ohio'))))
```
What states does the Ohio run through?

(b) GEOQUERY

Figure 1: Sample meaning representations

aspects of PHARAOH's phrase-based model can be used to improve WASP^{-1}, resulting in a hybrid system whose overall performance is the best.

2 MRLs and Test Domains

In this work, we consider input MRs with a hierarchical structure similar to Moore (2002). The only restriction on the MRL is that it be defined by an available unambiguous context-free grammar (CFG), which is true for almost all computer languages. We also assume that the order in which MR predicates appear is relevant, i.e. the order can affect the meaning of the MR. Note that the order in which predicates appear need not be the same as the word order of the target NL, and therefore, the content planner need not know about the target NL grammar (Shieber, 1993).

To ground our discussion, we consider two application domains which were originally used to demonstrate semantic parsing. The first domain is ROBOCUP. In the ROBOCUP Coach Competition (`www.robocup.org`), teams of agents compete in a simulated soccer game and receive coach advice written in a formal language called CLANG (Chen et al., 2003). The task is to build a system that translates this formal advice into English. Figure 1(a) shows a piece of sample advice.

The second domain is GEOQUERY, where a functional, variable-free query language is used for querying a small database on U.S. geography (Kate et al., 2005). The task is to translate formal queries into NL. Figure 1(b) shows a sample query.

3 Generation using SMT Methods

In this section, we show how SMT methods can be used to construct a tactical generator. This is in contrast to existing work that focuses on the use of NLG in interlingual MT (Whitelock, 1992), in which the roles of NLG and MT are switched. We first consider using a phrase-based SMT system, PHARAOH, for NLG. Then we show how to invert an SMT-based semantic parser, WASP, to produce a more effective generation system.

3.1 Generation using PHARAOH

PHARAOH (Koehn et al., 2003) is an SMT system that uses phrases as basic translation units. During decoding, the source sentence is segmented into a sequence of phrases. These phrases are then reordered and translated into phrases in the target language, which are joined together to form the output sentence. Compared to earlier word-based methods such as IBM Models (Brown et al., 1993), phrase-based methods such as PHARAOH are much more effective in producing idiomatic translations, and are currently the best performing methods in SMT (Koehn and Monz, 2006).

To use PHARAOH for NLG, we simply treat the source MRL as an NL, so that phrases in the MRL are sequences of MR tokens. Note that the grammaticality of MRs is not an issue here, as they are given as input.

3.2 WASP: The Semantic Parsing Algorithm

Before showing how generation can be performed by inverting a semantic parser, we present a brief overview of WASP (Wong and Mooney, 2006), the SMT-based semantic parser on which this work is based.

To describe WASP, it is best to start with an example. Consider the task of translating the English sentence in Figure 1(a) into CLANG. To do this, we may first generate a parse tree of the input sentence. The meaning of the sentence is then obtained by combining the meanings of the phrases. This process can be formalized using a *synchronous context-free grammar* (SCFG), originally developed as a grammar formalism that combines syntax analysis and code generation in compilers (Aho and Ullman, 1972). It has been used in syntax-based SMT to model the translation of one NL to another (Chiang, 2005). A derivation for a SCFG gives rise to multiple isomorphic parse trees. Figure 2 shows a partial parse of the sample sentence and its corre-

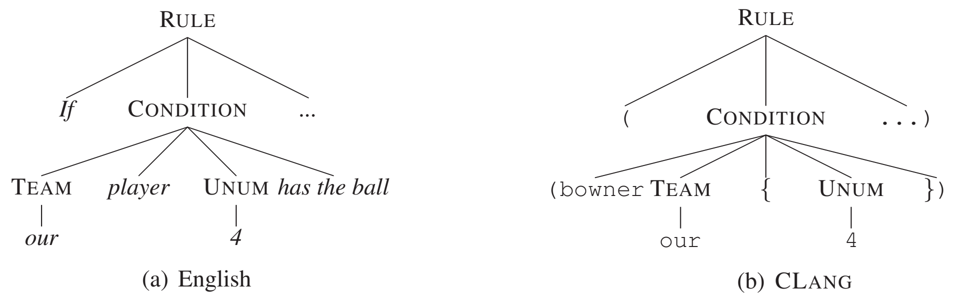


Figure 2: Partial parse trees for the CLANG statement and its English gloss shown in Figure 1(a)

sponding CLANG parse from which an MR is constructed. Note that the two parse trees are isomorphic (ignoring terminals).

Each SCFG rule consists of a non-terminal, X, on the left-hand side (LHS), and a pair of strings, $\langle \alpha, \beta \rangle$, on the right-hand side (RHS). The non-terminals in β are a permutation of the non-terminals in α (indices are used to show their correspondence). In WASP, α denotes an NL phrase, and $X \to \beta$ is a production of the MRL grammar. Below are the SCFG rules that generate the parses in Figure 2:

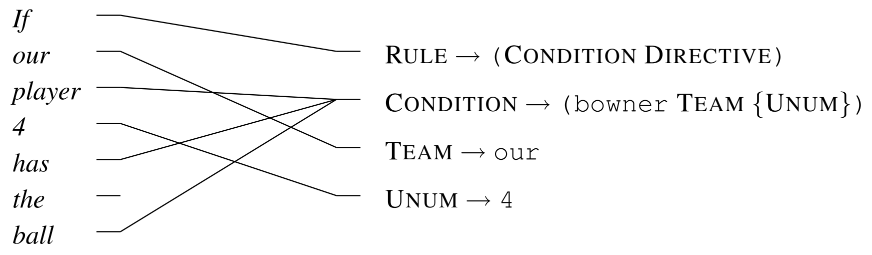
$$\begin{aligned}\text{RULE} &\to \langle \textit{if}\ \text{CONDITION}_{\boxed{1}}\ ,\ \text{DIRECTIVE}_{\boxed{2}}\ .\ ,\\ &\quad\ \texttt{(}\text{CONDITION}_{\boxed{1}}\ \text{DIRECTIVE}_{\boxed{2}}\texttt{)} \rangle\\ \text{CONDITION} &\to \langle \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{has the}\\ &\quad\ \textit{ball}\ ,\ \texttt{(bowner}\ \text{TEAM}_{\boxed{1}}\ \{\text{UNUM}_{\boxed{2}}\}\texttt{)} \rangle\\ \text{TEAM} &\to \langle \textit{our}\ ,\ \texttt{our} \rangle\\ \text{UNUM} &\to \langle \textit{4}\ ,\ \texttt{4} \rangle\end{aligned}$$

All derivations start with a pair of co-indexed start symbols of the MRL grammar, $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$, and each step involves the rewriting of a pair of co-indexed non-terminals (by α and β, respectively). Given an input sentence, $\mathbf{e}$, the task of semantic parsing is to find a derivation that yields $\langle \mathbf{e}, \mathbf{f} \rangle$, so that $\mathbf{f}$ is an MR translation of $\mathbf{e}$.

Parsing with WASP requires a set of SCFG rules. These rules are learned using a word alignment model, which finds an optimal mapping from words to MR predicates given a set of training sentences and their correct MRs. Word alignment models have been widely used for lexical acquisition in SMT (Brown et al., 1993; Koehn et al., 2003). To use a word alignment model in the semantic parsing scenario, we can treat the MRL simply as an NL, and MR tokens as words, but this often leads to poor results. First, not all MR tokens carry specific meanings. For example, in CLANG, parentheses and braces are delimiters that are semantically vacuous. Such tokens can easily confuse the word alignment model. Second, MR tokens may exhibit polysemy. For example, the CLANG predicate `pt` has three meanings based on the types of arguments it is given (Chen et al., 2003). Judging from the `pt` token alone, the word alignment model would not be able to identify its exact meaning.

A simple, principled way to avoid these difficulties is to represent an MR using a list of productions used to generate it. This list is used in lieu of the MR in a word alignment. Figure 3 shows an example. Here the list of productions corresponds to the top-down, left-most derivation of an MR. For each MR there is a unique linearized parse-tree, since the MRL grammar is unambiguous. Note that the structure of the parse tree is preserved through linearization. This allows us to extract SCFG rules in a bottom-up manner, assuming the alignment is n-to-1 (each word is linked to at most one production). Extraction starts with productions whose RHS is all terminals, followed by those with non-terminals. (Details can be found in Wong and Mooney (2006).) The rules extracted from Figure 3 would be almost the same as those used in Figure 2, except the one for `bowner`: CONDITION $\to \langle$TEAM$_{\boxed{1}}$ *player* UNUM$_{\boxed{2}}$ *has* (1) *ball*, `(bowner` TEAM$_{\boxed{1}}$ {UNUM$_{\boxed{2}}$}`)`$\rangle$. The token (1) denotes a *word gap* of size 1, due to the unaligned word *the* that comes between *has* and *ball*. It can be seen as a non-terminal that expands to at most one word, allowing for some flexibility in pattern matching.

In WASP, GIZA++ (Och and Ney, 2003) is used to obtain the best alignments from the training examples. Then SCFG rules are extracted from these alignments. The resulting SCFG, however, can be



If
our
player
4
has
the
ball
RULE → (CONDITION DIRECTIVE)
CONDITION → (bowner TEAM {UNUM})
TEAM → our
UNUM → 4

Figure 3: Partial word alignment for the CLANG statement and its English gloss shown in Figure 1(a)

ambiguous. Therefore, a maximum-entropy model that defines the conditional probability of derivations ($\mathbf{d}$) given an input sentence ($\mathbf{e}$) is used for disambiguation:

$$\Pr_\lambda(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\lambda(\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d}) \qquad (1)$$

The feature functions, f_i, are the number of times each rule is used in a derivation. $Z_\lambda(\mathbf{e})$ is the normalizing factor. The model parameters, λ_i, are trained using L-BFGS (Nocedal, 1980) to maximize the conditional log-likelihood of the training examples (with a Gaussian prior). The decoding task is thus to find a derivation $\mathbf{d}^\star$ that maximizes $\Pr_\lambda(\mathbf{d}^\star|\mathbf{e})$, and the output MR translation, $\mathbf{f}^\star$, is the yield of $\mathbf{d}^\star$. This can be done in cubic time with respect to the length of $\mathbf{e}$ using an Earley chart parser.

3.3 Generation by Inverting WASP

Now we show how to invert WASP to produce WASP^{-1}, and use it for NLG. We can use the same grammar for both parsing and generation, a particularly appealing aspect of using WASP. Since an SCFG is fully symmetric with respect to both generated strings, the same chart used for parsing can be easily adapted for efficient generation (Shieber, 1988; Kay, 1996).

Given an input MR, $\mathbf{f}$, WASP^{-1} finds a sentence $\mathbf{e}$ that maximizes $\Pr(\mathbf{e}|\mathbf{f})$. It is difficult to directly model $\Pr(\mathbf{e}|\mathbf{f})$, however, because it has to assign low probabilities to output sentences that are not grammatical. There is no such requirement for parsing, because the use of the MRL grammar ensures the grammaticality of all output MRs. For generation, we need an NL grammar to ensure grammaticality, but this is not available *a priori*.

This motivates the *noisy-channel model* for WASP^{-1}, where $\Pr(\mathbf{e}|\mathbf{f})$ is divided into two smaller components:

$$\arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \qquad (2)$$

$\Pr(\mathbf{e})$ is the *language model*, and $\Pr(\mathbf{f}|\mathbf{e})$ is the *parsing model*. The generation task is to find a sentence $\mathbf{e}$ such that (1) $\mathbf{e}$ is a good sentence a priori, and (2) its meaning is the same as the input MR. For the language model, we use an n-gram model, which is remarkably useful in ranking candidate generated sentences (Knight and Hatzivassiloglou, 1995; Bangalore et al., 2000; Langkilde-Geary, 2002). For the parsing model, we re-use the one from WASP (Equation 1). Hence computing (2) means maximizing the following:

$$\begin{aligned} & \max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \\ \approx\ & \max_{\mathbf{d} \in D(\mathbf{f})} \Pr(\mathbf{e}(\mathbf{d})) \Pr_\lambda(\mathbf{d}|\mathbf{e}(\mathbf{d})) \\ =\ & \max_{\mathbf{d} \in D(\mathbf{f})} \frac{\Pr(\mathbf{e}(\mathbf{d})) \cdot \exp \sum_i \lambda_i f_i(\mathbf{d})}{Z_\lambda(\mathbf{e}(\mathbf{d}))} \qquad (3) \end{aligned}$$

where $D(\mathbf{f})$ is the set of derivations that are consistent with $\mathbf{f}$, and $\mathbf{e}(\mathbf{d})$ is the output sentence that a derivation $\mathbf{d}$ yields. Compared to most existing work on generation, WASP^{-1} has the following characteristics:

1. It does not require any lexical information in the input MR, so lexical selection is an integral part of the decoding algorithm.

2. Each predicate is translated to a phrase. Moreover, it need not be a contiguous phrase (consider the SCFG rule for `bowner` in Section 3.2).

For decoding, we use an Earley chart generator that scans the input MR from left to right. This implies that each chart item covers a certain substring of the input MR, not a subsequence in general. It requires the order in which MR predicates appear to be fixed, i.e. the order determines the meaning of the MR. Since the order need not be identical to the word order of the target NL, there is no need for the content planner to know the target NL grammar, which is learned from the training data.

Overall, the noisy-channel model is a weighted SCFG, obtained by intersecting the NL side of the WASP SCFG with the n-gram language model. The chart generator is very similar to the chart parser, except for the following:

1. To facilitate the calculation of $\Pr(\mathbf{e}(\mathbf{d}))$, chart items now include a list of $(n-1)$-grams that encode the context in which output NL phrases appear. The size of the list is $2N+2$, where N is the number of non-terminals to be rewritten in the dotted rule.

2. Words are generated from word gaps through special rules $(g) \rightarrow \langle \alpha, \emptyset \rangle$, where the word gap, (g), is treated as a non-terminal, and α is the NL string that fills the gap ($|\alpha| \leq g$). The empty set symbol indicates that the NL string does not carry any meaning. There are similar constructs in Carroll et al. (1999) that generate function words. Furthermore, to improve efficiency, our generator only considers gap fillers that have been observed during training.

3. The normalizing factor in (3), $Z_\lambda(\mathbf{e}(\mathbf{d}))$, is not a constant and varies across the output string, $\mathbf{e}(\mathbf{d})$. (Note that $Z_\lambda(\mathbf{e})$ is fixed for parsing.) This is unfortunate because the calculation of $Z_\lambda(\mathbf{e}(\mathbf{d}))$ is expensive, and it is not easy to incorporate it into the chart generation algorithm. Normalization is done as follows. First, compute the k-best candidate output strings based on the unnormalized version of (3), $\Pr(\mathbf{e}(\mathbf{d})) \cdot \exp \sum_i \lambda_i f_i(\mathbf{d})$. Then re-rank the list by normalizing the scores using $Z_\lambda(\mathbf{e}(\mathbf{d}))$, which is obtained by running the inside-outside algorithm on each output string. This results in a decoding algorithm that is approximate—the best output string might not be in the k-best list—and takes cubic time with respect to the length of *each* of the k candidate output strings ($k = 100$ in our experiments).

Learning in WASP^{-1} involves two steps. First, a back-off n-gram language model with Good-Turing discounting and no lexical classes[2] is built from all training sentences using the SRILM Toolkit (Stolcke, 2002). We use $n = 2$ since higher values seemed to cause overfitting in our domains. Next, the parsing model is trained as described in Section 3.2.

[2]This is to ensure that the same language model is used in all systems that we tested.

4 Improving the SMT-based Generators

The SMT-based generation algorithms, PHARAOH and WASP^{-1}, while reasonably effective, can be substantially improved by borrowing ideas from each other.

4.1 Improving the PHARAOH-based Generator

A major weakness of PHARAOH as an NLG system is its inability to exploit the formal structure of the MRL. Like WASP^{-1}, the phrase extraction algorithm of PHARAOH is based on the output of a word alignment model such as GIZA++ (Koehn et al., 2003), which performs poorly when applied directly to MRLs (Section 3.2).

We can improve the PHARAOH-based generator by supplying linearized parse-trees as input rather than flat MRs. As a result, the basic translation units are sequences of MRL productions, rather than sequences of MR tokens. This way PHARAOH can exploit the formal grammar of the MRL to produce high-quality phrase pairs. The same idea is used in WASP^{-1} to produce high-quality SCFG rules. We call the resulting hybrid NLG system PHARAOH++.

4.2 Improving the WASP-based Generator

There are several aspects of PHARAOH that can be used to improve WASP^{-1}. First, the probabilistic model of WASP^{-1} is less than ideal as it requires an extra re-ranking step for normalization, which is expensive and prone to over-pruning. To remedy this situation, we can borrow the probabilistic model of PHARAOH, and define the parsing model as:

$$\Pr(\mathbf{d}|\mathbf{e}(\mathbf{d})) = \prod_{d \in \mathbf{d}} w(r(d)) \tag{4}$$

which is the product of the weights of the rules used in a derivation $\mathbf{d}$. The rule weight, $w(X \rightarrow \langle \alpha, \beta \rangle)$, is in turn defined as:

$$P(\beta|\alpha)^{\lambda_1} P(\alpha|\beta)^{\lambda_2} P_w(\beta|\alpha)^{\lambda_3} P_w(\alpha|\beta)^{\lambda_4} \exp(-|\alpha|)^{\lambda_5}$$

where $P(\beta|\alpha)$ and $P(\alpha|\beta)$ are the relative frequencies of β and α, and $P_w(\beta|\alpha)$ and $P_w(\alpha|\beta)$ are the lexical weights (Koehn et al., 2003). The word penalty, $\exp(-|\alpha|)$, allows some control over the output sentence length. Together with the language model, the new formulation of $\Pr(\mathbf{e}|\mathbf{f})$ is a log-linear model with λ_i as parameters. The advantage of this model is that maximization requires no normalization and can be done exactly and efficiently. The model parameters are trained using minimum error-rate training (Och, 2003).

Following the phrase extraction phase in PHARAOH, we eliminate word gaps by incorporating unaligned words as part of the extracted NL phrases (Koehn et al., 2003). The reason is that while word gaps are useful in dealing with unknown phrases during semantic parsing, for generation, using *known* phrases generally leads to better fluency. For the same reason, we also allow the extraction of longer phrases that correspond to multiple predicates (but no more than 5).

We call the resulting hybrid system WASP^{-1}++. It is similar to the syntax-based SMT system of Chiang (2005), which uses both SCFG and PHARAOH's probabilistic model. The main difference is that we use the MRL grammar to constrain rule extraction, so that significantly fewer rules are extracted, making it possible to do exact inference.

5 Experiments

We evaluated all four SMT-based NLG systems introduced in this paper: PHARAOH, WASP^{-1}, and the hybrid systems, PHARAOH++ and WASP^{-1}++.

We used the ROBOCUP and GEOQUERY corpora in our experiments. The ROBOCUP corpus consists of 300 pieces of coach advice taken from the log files of the 2003 ROBOCUP Coach Competition. The advice was written in CLANG and manually translated to English (Kuhlmann et al., 2004). The average MR length is 29.47 tokens, or 12.82 nodes for linearized parse-trees. The average sentence length is 22.52. The GEOQUERY corpus consists of 880 English questions gathered from various sources. The questions were manually translated to the functional GEOQUERY language (Kate et al., 2005). The average MR length is 17.55 tokens, or 5.55 nodes for linearized parse-trees. The average sentence length is 7.57.

Reference: *If our player 2, 3, 7 or 5 has the ball and the ball is close to our goal line ...*
PHARAOH++: *If player 3 has the ball is in 2 5 the ball is in the area near our goal line ...*
WASP^{-1}++: *If players 2, 3, 7 and 5 has the ball and the ball is near our goal line ...*

Figure 4: Sample partial system output in the ROBOCUP domain

	ROBOCUP		GEOQUERY	
	BLEU	NIST	BLEU	NIST
PHARAOH	0.3247	5.0263	0.2070	3.1478
WASP^{-1}	0.4357	5.4486	0.4582	5.9900
PHARAOH++	0.4336	5.9185	**0.5354**	6.3637
WASP^{-1}++	**0.6022**	**6.8976**	**0.5370**	**6.4808**

Table 1: Results of automatic evaluation; **bold** type indicates the best performing system (or systems) for a given domain-metric pair ($p < 0.05$)

5.1 Automatic Evaluation

We performed 4 runs of 10-fold cross validation, and measured the performance of the learned generators using the BLEU score (Papineni et al., 2002) and the NIST score (Doddington, 2002). Both MT metrics measure the precision of a translation in terms of the proportion of n-grams that it shares with the reference translations, with the NIST score focusing more on n-grams that are less frequent and more informative. Both metrics have recently been used to evaluate generators (Langkilde-Geary, 2002; Nakanishi et al., 2005; Belz and Reiter, 2006).

All systems were able to generate sentences for more than 97% of the input. Figure 4 shows some sample output of the systems. Table 1 shows the automatic evaluation results. Paired t-tests were used to measure statistical significance. A few observations can be made. First, WASP^{-1} produced a more accurate generator than PHARAOH. Second, PHARAOH++ significantly outperformed PHARAOH, showing the importance of exploiting the formal structure of the MRL. Third, WASP^{-1}++ significantly outperformed WASP^{-1}. Most of the gain came from PHARAOH's probabilistic model. Decoding was also 4–11 times faster, despite exact inference and a larger grammar due to extraction of longer phrases. Lastly, WASP^{-1}++ significantly outperformed PHARAOH++ in the ROBOCUP

	ROBOCUP		GEOQUERY	
	Flu.	*Ade.*	*Flu.*	*Ade.*
PHARAOH++	2.5	2.9	**4.3**	**4.7**
WASP^{-1}++	**3.6**	**4.0**	**4.1**	**4.7**

Table 2: Results of human evaluation

domain. This is because WASP^{-1}++ allows discontiguous NL phrases and PHARAOH++ does not. Such phrases are commonly used in ROBOCUP for constructions like: **players** *2* **,** *3* **,** *7* **and** *5*; 26.96% of the phrases generated during testing were discontiguous. When faced with such predicates, PHARAOH++ would consistently omit some of the words: e.g. **players** *2 3 7 5*, or not learn any phrases for those predicates at all. On the other hand, only 4.47% of the phrases generated during testing for GEOQUERY were discontiguous, so the advantage of WASP^{-1}++ over PHARAOH++ was not as obvious.

Our BLEU scores are not as high as those reported in Langkilde-Geary (2002) and Nakanishi et al. (2005), which are around 0.7–0.9. However, their work involves the regeneration of automatically parsed text, and the MRs that they use, which are essentially dependency parses, contain extensive lexical information of the target NL.

5.2 Human Evaluation

Automatic evaluation is only an imperfect substitute for human assessment. While it is found that BLEU and NIST correlate quite well with human judgments in evaluating NLG systems (Belz and Reiter, 2006), it is best to support these figures with human evaluation, which we did on a small scale. We recruited 4 native speakers of English with no previous experience with the ROBOCUP and GEOQUERY domains. Each subject was given the same 20 sentences for each domain, randomly chosen from the test sets. For each sentence, the subjects were asked to judge the output of PHARAOH++ and WASP^{-1}++ in terms of fluency and adequacy. They were presented with the following definition, adapted from Koehn and Monz (2006):

Score	*Fluency*	*Adequacy*
5	Flawless English	All meaning
4	Good English	Most meaning
3	Non-native English	Some meaning
2	Disfluent English	Little meaning
1	Incomprehensible	No meaning

For each generated sentence, we computed the average of the 4 human judges' scores. No score normalization was performed. Then we compared the two systems using a paired t-test. Table 2 shows that WASP^{-1}++ produced better generators than PHARAOH++ in the ROBOCUP domain, consistent with the results of automatic evaluation.

	PHARAOH++		WASP^{-1}++	
	BLEU	NIST	BLEU	NIST
English	0.5344	5.3289	**0.6035**	**5.7133**
Spanish	**0.6042**	**5.6321**	**0.6175**	**5.7293**
Japanese	0.6171	**4.5357**	**0.6585**	**4.6648**
Turkish	0.4562	**4.2220**	**0.4824**	**4.3283**

Table 3: Results of automatic evaluation on the multilingual GEOQUERY data set

5.3 Multilingual Experiments

Lastly, we describe our experiments on the multilingual GEOQUERY data set. The 250-example data set is a subset of the larger GEOQUERY corpus. All English questions in this data set were manually translated into Spanish, Japanese and Turkish, while the corresponding MRs remain unchanged. Table 3 shows the results, which are similar to previous results on the larger GEOQUERY corpus. WASP^{-1}++ outperformed PHARAOH++ for some language-metric pairs, but otherwise performed comparably.

6 Related Work

Numerous efforts have been made to unify the tasks of semantic parsing and tactical generation. One of the earliest espousals of the notion of grammar reversability can be found in Kay (1975). Shieber (1988) further noted that not only a single grammar can be used for parsing and generation, but the same language-processing architecture can be used for both tasks. Kay (1996) identified parsing charts as such an architecture, which led to the development of various chart generation algorithms: Carroll et al. (1999) for HPSG, Bangalore et al. (2000) for LTAG, Moore (2002) for unification grammars, White and Baldridge (2003) for CCG. More recently, statistical chart generators have emerged, including White (2004) for CCG, Carroll and Oepen (2005) and Nakanishi et al. (2005) for HPSG. Many of these systems, however, focus on the task of *surface realization*—inflecting and ordering words—which ignores the problem of lexical selection. In contrast, our SMT-based methods integrate lexical selection and realization in an elegant framework and automatically learn all of their linguistic knowledge from an annotated corpus.

7 Conclusion

We have presented four tactical generation systems based on various SMT-based methods. In particular, the hybrid system produced by inverting the WASP semantic parser shows the best overall results across different application domains.

Acknowledgments

We would like to thank Kevin Knight, Jason Baldridge, Razvan Bunescu, and the anonymous reviewers for their valuable comments. We also sincerely thank the four annotators who helped us evaluate the SMT-based generators. This research was supported by DARPA under grant HR0011-04-1-0007 and a gift from Google Inc.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

S. Bangalore, O. Rambow, and S. Whittaker. 2000. Evaluation metrics for generation. In *Proc. INLG-00*, pages 1–8, Mitzpe Ramon, Israel, July.

A. Belz and E. Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proc. EACL-06*, pages 313–320, Trento, Italy, April.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

J. Carroll and S. Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proc. IJCNLP-05*, pages 165–176, Jeju Island, Korea, October.

J. Carroll, A. Copestake, D. Flickinger, and V. Poznański. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proc. EWNLG-99*, pages 86–95, Toulouse, France.

M. Chen et al. 2003. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL-05*, pages 263–270, Ann Arbor, MI, June.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. ARPA Workshop on Human Language Technology*, pages 128–132, San Diego, CA.

U. Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proc. HLT/NAACL-03*, Edmonton, Canada.

P. S. Jacobs. 1985. PHRED: A generator for natural language interfaces. *Computational Linguistics*, 11(4):219–242.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proc. AAAI-05*, pages 1062–1068, Pittsburgh, PA, July.

M. Kay. 1975. Syntactic processing and functional sentence perspective. In *Theoretical Issues in Natural Language Processing—Supplement to the Proceedings*, pages 12–15, Cambridge, MA, June.

M. Kay. 1996. Chart generation. In *Proc. ACL-96*, pages 200–204, San Francisco, CA.

K. Knight and V. Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proc. ACL-95*, pages 252–260, Cambridge, MA.

P. Koehn and C. Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *Proc. SMT-06 Workshop*, pages 102–121, New York City, NY, June.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT/NAACL-03*, Edmonton, Canada.

G. Kuhlmann, P. Stone, R. J. Mooney, and J. W. Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proc. of the AAAI-04 Workshop on Supervisory Control of Learning and Adaptive Systems*, San Jose, CA, July.

I. Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proc. INLG-02*, pages 17–24, Harriman, NY, July.

R. C. Moore. 2002. A complete, efficient sentence-realization algorithm for unification grammar. In *Proc. INLG-02*, pages 41–48, Harriman, NY, July.

H. Nakanishi, Y. Miyao, and J. Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proc. IWPT-05*, pages 93–102, Vancouver, Canada, October.

J. Nocedal. 1980. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, July.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL-03*, pages 160–167, Sapporo, Japan, July.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL-02*, pages 311–318, Philadelphia, PA, July.

S. M. Shieber. 1988. A uniform architecture for parsing and generation. In *Proc. COLING-88*, pages 614–619, Budapest, Hungary.

S. M. Shieber. 1993. The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190.

A. Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *Proc. ICSLP-02*, pages 901–904, Denver, CO.

M. White and J. Baldridge. 2003. Adapting chart realization to CCG. In *Proc. EWNLG-03*, Budapest, Hungary, April.

M. White. 2004. Reining in CCG chart realization. In *Proc. INLG-04*, New Forest, UK, July.

P. Whitelock. 1992. Shake-and-bake translation. In *Proc. COLING-92*, pages 784–791, Nantes, France.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. HLT/NAACL-06*, pages 439–446, New York City, NY, June.

Lexicalized Markov Grammars for Sentence Compression[*]

Michel Galley and **Kathleen R. McKeown**
Columbia University
Department of Computer Science
New York, NY 10027, USA
{galley,kathy}@cs.columbia.edu

Abstract

We present a sentence compression system based on synchronous context-free grammars (SCFG), following the successful noisy-channel approach of (Knight and Marcu, 2000). We define a head-driven Markovization formulation of SCFG deletion rules, which allows us to lexicalize probabilities of constituent deletions. We also use a robust approach for tree-to-tree alignment between arbitrary document-abstract parallel corpora, which lets us train lexicalized models with much more data than previous approaches relying exclusively on scarcely available document-compression corpora. Finally, we evaluate different Markovized models, and find that our selected best model is one that exploits head-modifier bilexicalization to accurately distinguish adjuncts from complements, and that produces sentences that were judged more grammatical than those generated by previous work.

1 Introduction

Sentence compression addresses the problem of removing words or phrases that are not necessary in the generated output of, for instance, summarization and question answering systems. Given the need to ensure grammatical sentences, a number of researchers have used syntax-directed approaches that perform transformations on the output of syntactic parsers (Jing, 2000; Dorr et al., 2003). Some of them (Knight and Marcu, 2000; Turner and Charniak, 2005) take an empirical approach, relying on formalisms equivalent to probabilistic synchronous context-free grammars (SCFG) (Lewis and Stearns, 1968; Aho and Ullman, 1969) to extract compression rules from aligned Penn Treebank (PTB) trees. While their approach proved successful, their reliance on standard maximum likelihood estimators for SCFG productions results in considerable sparseness issues, especially given the relative flat structure of PTB trees; in practice, many SCFG productions are seen only once. This problem is exacerbated for the compression task, which has only scarce training material available.

In this paper, we present a head-driven Markovization of SCFG compression rules, an approach that was successfully used in syntactic parsing (Collins, 1999; Klein and Manning, 2003) to alleviate issues intrinsic to relative frequency estimation of treebank productions. Markovization for sentence compression provides several benefits, including the ability to condition deletions on a flexible amount of syntactic context, to treat head-modifier dependencies independently, and to lexicalize SCFG productions.

Another part of our effort focuses on better alignment models for extracting SCFG compression rules from parallel data, and to improve upon (Knight and Marcu, 2000), who could only exploit 1.75% of the Ziff-Davis corpus because of stringent assumptions about human abstractive behavior. To alleviate their restrictions, we rely on a robust approach for aligning trees of arbitrary document-abstract sentence pairs. After accounting for sentence pairs with both substitutions and deletions, we reached a retention of more than 25% of the Ziff-Davis data, which greatly benefited the lexical probabilities incorporated into our Markovized SCFGs.

Our work provides three main contributions:

[*]This material is based on research supported in part by the U.S. National Science Foundation (NSF) under Grant No. IIS-05-34871 and the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

(1) Our lexicalized head-driven Markovization yields more robust probability estimates, and our compressions outperform (Knight and Marcu, 2000) according to automatic and human evaluation. (2) We provide a comprehensive analysis of the impact of different Markov orders for sentence compression, similarly to a study done for PCFGs (Klein and Manning, 2003). (3) We provide a framework for exploiting document-abstract sentence pairs that are not purely compressive, and augment the available training resources for syntax-directed sentence compression systems.

2 Synchronous Grammars for Sentence Compression

One successful syntax-driven approach (Knight and Marcu, 2000, henceforth K&M) relies on synchronous context-free grammars (SCFG) (Lewis and Stearns, 1968; Aho and Ullman, 1969). SCFGs can be informally defined as context-free grammars (CFGs) whose productions have two right-hand side strings instead of one, namely *source* and *target* right-hand side. In the case of sentence compression, we restrict the target side to be a sub-sequence of the source side (possibly identical), and we will call this restricted grammar a *deletion SCFG*. For instance, a deletion SCFG rule that removes an adverbial phrase (ADVP) between an noun phrase (NP) and a verb phrase (VP) may be written as follows:

$$\text{S} \rightarrow \langle \text{NP ADVP VP, NP VP} \rangle$$

In a sentence compression framework similar to the one presented by K&M, we build SCFGs that are fully trainable from a corpus of document and reduced sentences. Such an approach comprises two subproblems: (1) transform tree pairs into synchronous grammar derivations; (2) based on these derivations, assign probabilities to deletion SCFG productions, and more generally, to compressions produced by such grammars. Since the main point of our paper lies in the exploration of better probability estimates through Markovization and lexicalization of SCFGs, we first address the latter problem, and discuss the task of building synchronous derivations only later in Section 4.

2.1 Stochastic Synchronous Grammars

The overall goal of a sentence compression system is to transform a given input sentence $\mathbf{f}$ into a concise and grammatical sentence $\mathbf{c} \in \mathbf{C}$, which is a subsequence of $\mathbf{f}$. Similarly to K&M and many successful syntactic parsers (Collins, 1999; Klein and Manning, 2003), our sentence compression system is *generative*, and attempts to find the optimal compression $\hat{\mathbf{c}}$ by estimating the following function:[1]

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in\mathbf{C}} \Big\{ p(\mathbf{c}|\mathbf{f}) \Big\} = \arg\max_{\mathbf{c}\in\mathbf{C}} \Big\{ p(\mathbf{f},\mathbf{c}) \Big\} \quad (1)$$

If $\tau(\mathbf{f},\mathbf{c})$ is the set of all tree pairs that yield $(\mathbf{f},\mathbf{c})$ according to some underlying SCFG, we can estimate the probability of the sentence pair using:

$$p(\mathbf{f},\mathbf{c}) = \sum_{(\pi_\mathbf{f},\pi_\mathbf{c})\in\tau(\mathbf{f},\mathbf{c})} P(\pi_\mathbf{f},\pi_\mathbf{c}) \quad (2)$$

We note that, in practice (and as in K&M), Equation 2 is often approximated by restricting $\tau(\mathbf{f},\mathbf{c})$ to a unique full tree $\hat{\pi}_\mathbf{f}$, the best hypothesis of an off-the-shelf syntactic parser. This implies that each possible compression $\mathbf{c}$ is the target-side yield of at most one SCFG derivation.

As in standard PCFG history-based models, the probability of the entire structure (Equation 2) is factored into probabilities of grammar productions. If θ is a derivation $\theta = r^1 \circ \cdots \circ r^j \cdots \circ r^J$, where r^j denotes the SCFG rule $l^j \rightarrow \langle \alpha_f^j, \alpha_c^j \rangle$, we get:

$$p(\pi_\mathbf{f},\pi_\mathbf{c}) = \prod_{j=1}^{J} p(\alpha_f^j, \alpha_c^j | l^j) \quad (3)$$

The question we will now address is how to estimate the probability $p(\alpha_f^j, \alpha_c^j | l^j)$ of each SCFG production.

2.2 Lexicalized Head-Driven Markovization of Synchronous Grammars

A main issue in our enterprise is to reliably estimate productions of deletion SCFGs. In a sentence compression framework as the one presented by K&M, we use aligned trees of the form of the Penn Treebank (PTB) (Marcus et al., 1994) to acquire and score SCFG productions. However, the use of the PTB structure faces many challenges also encountered in probabilistic parsing.

[1] In their noisy-channel approach, K&M further break down $p(\mathbf{c},\mathbf{f})$ into $p(\mathbf{f}|\mathbf{c}) \cdot p(\mathbf{c})$, which we refrain from doing for reasons that will become obvious later.

Firstly, PTB tree structures are relatively flat, particularly within noun phrases. For instance, adjective phrases (ADJP)—which are generally good candidates for deletions—appear in 90 different NP-rooted SCFG productions in Ziff-Davis,[2] 61 of which appear only once, e.g., NP $\rightarrow \langle$DT ADJP JJ NN NN, DT JJ NN NN$\rangle$. While it may seem advantageous to maintain many constituents within the same domain of locality of an SCFG production, as we may hope to exploit its large syntactic context to condition deletions more accurately, the sparsity of such productions make them poor candidates for relative frequency estimation, especially in a task with limited quantities of training material. Indeed, our base training corpus described in Section 4 contains only 951 SCFG productions, 593 appearing once.

Secondly, syntactic categories in the PTB are particularly coarse grained, and lead to many incorrect context-free assumptions. Some important distinctions, such as between arguments and adjuncts, are beyond the scope of the PTB annotation, and it is often difficult to determine out of context whether a given constituent can safely be deleted from a right-hand side.

One first type of annotation that can effectively be added to each syntactic category is its lexical head and head part-of-speech (POS), following work in syntactic parsing (Collins, 1999). This type of annotation is particular beneficial in the case of, e.g., prepositional phrases (PP), which may be either complement or adjunct. As in the case of Figure 1 (in which adjuncts appear in italic), knowing that the PP headed by "from" appears in a VP headed by "fell" helps us to determine that the PP is a complement to the verb "fell", and that it should presumably not be deleted. Conversely, the PP headed by "because" modifying the same verb is an adjunct, and can safely be deleted if unimportant.[3] Also, as discussed in (Klein and Manning, 2003), POS annotation can be useful as a means of backing off to more frequently occurring head-modifier POS occurrences (e.g., VBD-IN) when specific bilexical co-

[2]Details about the SCFG extraction procedure are given in Section 4. In short, we refer here to a grammar generated from 823 sentence pairs.

[3]The PP headed by "from" is an optional argument, and thus may still be deleted. Our point is that lexical information in general should help give lower scores to deletions of constituents that are grammatically more prominent.

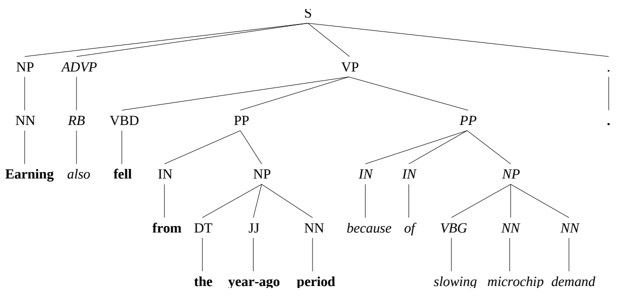


Figure 1: Penn Treebank tree with adjuncts in italic.

occurrences are sparsely seen (e.g., "fell"-"from"). At a lower level, lexicalization is clearly desirable for pre-terminals. Indeed, current SCFG models such as K&M have no direct way of preventing highly improbable single word removals, such as deletions of adverbs "never" or "nowhere", which may turn a negative statement into a positive one.[4]

A second type of annotation that can be added to syntactic categories is the so-called *parent annotation* (Johnson, 1998), which was effectively used in syntactic parsing to break unreasonable context-free assumptions. For instance, a PP with a VP parent is marked as PP^VP. It is reasonable to assume that, e.g., that constituents deep inside a PP have more chances to be removed than otherwise expected, and one may seek to increase the amount of vertical context that is available for conditioning each constituent deletion.

To achieve the above desiderata for better SCFG probability estimates—i.e., reduce the amount of sister annotation within each SCFG production, by conditioning deletions on a context smaller than an entire right-hand side, and at the same time increase the amount of ancestor and descendent annotation through parent (or ancestor) annotation and lexicalization—we follow the approach of (Collins, 1999; Klein and Manning, 2003), i.e., factorize n-ary grammar productions into products of n right-hand side probabilities, a technique sometimes called *Markovization*.

Markovization is generally head-driven, i.e., reflects a decomposition centered around the head of each CFG production:

$$l \rightarrow \Delta L^m \cdots L^1 H R^1 \cdots R^n \Delta \qquad (4)$$

[4]K&M incorporate lexical probabilities through n-gram models, but such language models are obviously not good for preventing such unreasonable deletions.

where H is the head, $L^1, \ldots, L^m$ the left modifiers, $R^1, \ldots, R^n$ are right modifiers, and Δ termination symbols needed for accurate probability estimations (e.g., to capture the fact that certain constituents are more likely than others to be the rightmost constituent); for simplicity, we will ignore Δ in later discussions. For a given SCFG production $l \rightarrow \langle \alpha_f, \alpha_c \rangle$, we ask, given the source RHS α_f that is assumed given (e.g., provided by a syntactic parser), which of its RHS elements are also present in α_c. That is, we write:

$$p(\alpha_c|\alpha_f, l) = \qquad (5)$$
$$p(k_l^m, \cdots, k_l^1, k_h, k_r^1, \cdots, k_r^n|\alpha_f, l)$$

where k_h, k_l^i, k_r^j ('k' for keep) are binary variables that are true if and only if constituents H, L_i, R^j (respectively) of the source RHS α_f are present in the target side α_c. Note that the conditional probability in Equation 5 enables us to estimate Equation 3, since $p(\alpha_f, \alpha_c|l) = p(\alpha_c|\alpha_f, l) \cdot p(\alpha_f|l)$. We can rely on a state-of-the-art probabilistic parser to effectively compute either $p(\alpha_f|l)$ or the probability of the entire tree $\pi_\mathbf{f}$, and need not worry about estimating this term. In the case of sentence compression from the one-best hypothesis of the parser, we can ignore $p(\alpha_f|l)$ altogether, since $\pi_\mathbf{f}$ is the same for all compressions.

We can rewrite Equation 5 exactly using a head-driven infinite-horizon Markovization:

$$p(\alpha_c|\alpha_f, l) = p(k_h|\alpha_f, l) \qquad (6)$$
$$\cdot \prod_{i=1\ldots m} p(k_l^i|k_l^1, \cdots, k_l^{i-1}, k_h, \alpha_f, l)$$
$$\cdot \prod_{i=1\ldots n} p(k_r^i|k_r^1, \cdots, k_r^{i-1}, k_h, \Lambda, \alpha_f, l)$$

where $\Lambda = (k_l^1, \cdots, k_l^m)$ is a term needed by the chain rule. One key issue is to make linguistically plausible assumptions to determine which conditioning variables in the terms should be deleted. Following our discussion in the first part of this section, we may start by making an order-s Markov approximation centered around the head, i.e., we condition each binary variable (e.g., k_r^i) on a context of up to s sister constituents between the current constituent and the head (e.g., $(R^{i-s}, \ldots, R^i)$). In order to incorporate bilexical dependencies between the head and each modifier, we also condition all modifier probabilities on head variables H (and k_h). These assumptions are overall quite similar to the ones made in Markovized parsing models. If we assume that all other conditioning variables in Equation 6 are irrelevant, we write:

$$p(\alpha_c|\alpha_f, l) = p_h(k_h|H, l) \qquad (7)$$
$$\cdot \prod_{i=1\ldots m} p_l(k_l^i|L^{i-s}, ..., L^i, k_l^{i-s}, ..., k_l^{i-1}, H, k_h, l)$$
$$\cdot \prod_{i=1\ldots n} p_r(k_r^i|R^{i-s}, ..., R^i, k_r^{i-s}, ..., k_r^{i-1}, H, k_h, l)$$

Note that it is important to condition deletions on both constituent histories $(R^{i-s}, \ldots, R^i)$ and non-deletion histories $(k_r^{i-s}, \ldots, k_r^{i-1})$; otherwise we would be unable to perform deletions that must operate jointly, as in production S $\rightarrow \langle$ADVP COMMA NP VP, NP VP$\rangle$ (in which the ADVP should not be deleted without the comma). Without binary histories, we often observed superfluous punctuation symbols and dangling coordinate conjunctions appearing in our outputs.

Finally, we label l with an order-v ancestor annotation, e.g., for the VP in Figure 1, $l = \epsilon$ for $v = 0$, $l =$VP^S for $v = 2$, and so on. We also replace H and modifiers L^i and R^i by lexicalized entries, e.g., $H =$(VP,VBD,*fell*) and $R^i =$(PP,IN,*from*). Note that to estimate $p_l(k_l^i|\cdots)$, we only lexicalize L^i and H, and none of the other conditioning modifiers, since this would, of course, introduce too many conditioning variables (the same goes for $p_r(k_r^i|\cdots)$). The question of how much sister and vertical (s and v) context is needed for effective sentence compression, and whether to use lexical or POS annotation, will be evaluated in detail in Section 5.

3 The Data

To acquire SCFG productions, we used Ziff-Davis, a corpus of technical articles and human abstractive summaries. Articles and summaries are paired by document, so the first step was to perform sentence alignment. In the particular case of sentence compression, a simple approach is to just consider compression pairs (**f**,**c**), where **c** is a substring of **f**. K&M identified only 1,087 such paired sentences in the entire corpus, which represents a recall of 1.75%.

For our empirical evaluations, we split the data as follows: among the 1,055 sentences that were taken to train systems described in K&M, we selected the first 32 sentence pairs to be an auxiliary test corpus (for future work), the next 200 sentences to be our development corpus, and the remaining 823 to be our base training corpus (ZD-0), which will be augmented with additional data as explained in the next section. We feel it is important to use a relatively large development corpus, since we will provide in Section 5 detailed analyses of model selection on the development set (e.g., by evaluating different Markov structures), and we want these findings to be as significant as possible. Finally, we used the same test data as K&M for human evaluation purposes (32 sentence pairs).

4 Tree Alignment and Synchronous Grammar Inference

We now describe methods to train SCFG models from sentence pairs. Given a tree pair $(\mathbf{f}, \mathbf{c})$, whose respective parses $(\pi_{\mathbf{f}}, \pi_{\mathbf{c}})$ were generated by the parser described in (Charniak and Johnson, 2005), the goal is to transform the tree pair into SCFG derivations, in order to build relative frequency estimates for our Markovized models from observed SCFG productions. Clearly, the two trees may sometimes be structurally quite different (e.g., a given PP may attach to an NP in $\pi_{\mathbf{f}}$, while attaching to VP in $\pi_{\mathbf{c}}$), and it is not always possible to build an SCFG derivation given the constraints in $(\pi_{\mathbf{f}}, \pi_{\mathbf{c}})$. The approach taken by K&M is to analyze both trees and count an SCFG rule whenever two nodes are "deemed to correspond", i.e., roots are the same, and α_c is a sub-sequence of α_f. This leads to a quite restricted number of different productions on our base training set (ZD-0): 823 different productions were extracted, 593 of which appear only once. This first approach has serious limitations; the assumption that sentence compression appropriately models human abstractive data is particularly problematic. This considerably limits the amount of training data that can be exploited in Ziff-Davis (which contains overall more than 4,000 documents-abstract pairs), and this makes it very difficult to train lexicalized models.

An approach to slightly loosen this assumption is to consider document-abstract sentence pairs in which the condensed version contains one or more substitutions or insertions. Consider for example the tree pair in Figure 2: the two sentences are syntactically very close, but the substitution of "computer" with "unit" makes this sentence pair unusable in the framework presented in K&M. Arguably, there should be ways to exploit abstract sentences that are slightly reworded in addition to being compressed. To use sentence pairs with insertions and substitutions, we must find a way to align tree pairs in order to identify SCFG productions. More specifically, we must define a constituent alignment between the paired abstract and document sentences, which determine how the two trees are synchronized in a derivation. Obtaining this alignment is no trivial matter as the number of non-deleting edits increases. To address this, we synchronized tree pairs by finding the constituent alignment that minimizes the edit distance between the two trees, i.e., minimize the number of terminals and non-terminals insertions, substitutions and deletions.[5] While criteria

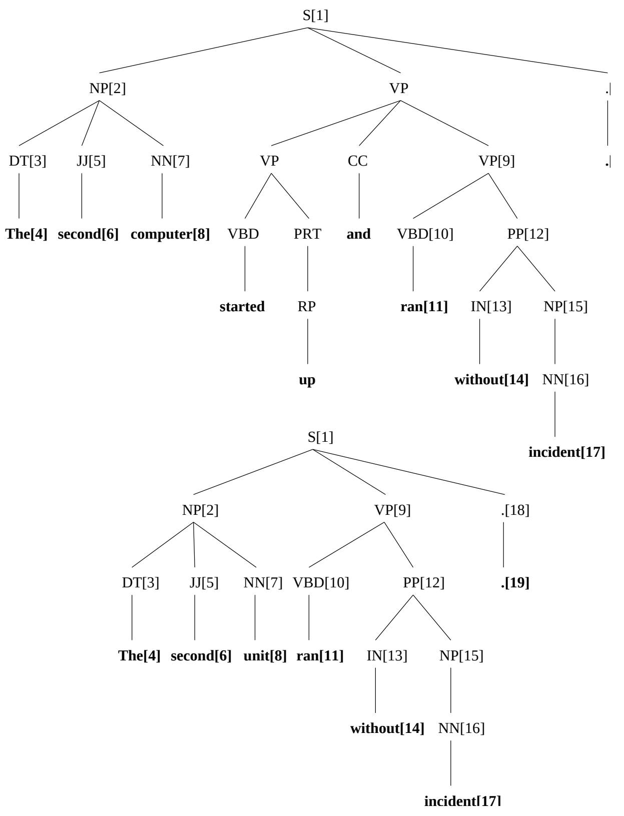


Figure 2: Full sentence and its revision. While the latter is not a compression of the former, it could still be used to gather statistics to train a sentence compression system, e.g., to learn the reduction of a VP coordination.

[5]The minimization problem is known to be NP hard, so we used an approximation algorithm (Zhang and Shasha, 1989) that

other than minimum tree edit distance may be effective, we found—after manual inspections of alignments between sentences with less than five non-deleting edits—that this method generally produces good alignments. A sample alignment is provided in Figure 2. Once a constituent alignment is available, it is then trivial to extract all deletion SCFG rules available in a tree pair, e.g., NP $\rightarrow \langle$DT JJ NN, DT JJ NN$\rangle$ in the figure.

We also exploited more general tree productions known as synchronous tree substitution grammar (STSG) rules, in an approach quite similar to (Turner and Charniak, 2005). For instance, the STSG rule rooted at S can be decomposed into two SCFG productions if we allow unary rules such as VP $\rightarrow$ VP to be freely added to the compressed tree. More specifically, we decompose any STSG rule that has in its target (compressed) RHS a single context free production, and that contains in its source (full) RHS a single context free production adjoined with any number of tree adjoining grammar (TAG) auxiliary trees (Joshi et al., 1975). In the figure, the initial tree is S $\rightarrow$ NP VP, and the adjoined (auxiliary) tree is VP $\rightarrow$ VP CC VP.[6] We found this approach quite helpful, since most useful compressions that mimic TAG adjoining operations are missed by the extraction procedure of K&M.

Since we found that exploiting sentence pairs containing insertions had adverse consequences in terms of compression accuracies, we only report experiments with sentence pairs containing no insertions. We gathered sentence pairs with up to six substitutions using minimum edit distance matching (we will refer to these sets as ZD-0 to ZD-6). With a limit of up to six substitutions (ZD-6), we were able to train our models on 16,787 sentences, which represents about 25% of the total number of summary sentences of the Ziff-Davis corpus.

5 Experiments

All experiments presented in this section are performed on the Ziff-Davis corpus. We note first that all probability estimates of our Markovized grammars are smoothed. Indeed, incorporating lexical dependencies within models trained on data sets as small as 16,000 sentence pairs would be quite futile without incorporating robust smoothing techniques. Different smoothing techniques were evaluated with our models, and we found that interpolated Witten-Bell discounting was the method that performed best. We used relative frequency estimates for each of the models presented in Section 2.2 (i.e., p_h, p_l, p_r), and trained p_l separately from p_r. We interpolated our most specific models (lexical heads, POS tags, ancestor and sister annotation) with lower-order models.[7]

Automatic evaluation on development sets is performed using word-level classification accuracy, i.e., the number of words correctly classified as being either deleted or not deleted, divided by the total number of words. In our first evaluation, we experimented with different horizontal and vertical Markovizations (Table 1). First, it appears that vertical annotation is moderately helpful. It provides gains in accuracy ranging from .5% to .9% for $v = 1$ over a simpler models ($v = 0$), but higher orders ($v > 1$) have a tendency to decrease performance. On the other hand, sister annotation of order 1 is much more critical, and provides 4.1% improvement over a simpler model ($s = 0, v = 0$). Manual examinations of compression outputs confirmed this analysis: without sister annotation, deletion of punctuation and function words (determiners, coordinate conjunctions, etc.) is often inaccurate, and compressions clearly lack fluency. This annotation is also helpful for phrasal deletions; for instance, we found that PPs are deleted in 31.4% of cases in Ziff-Davis if they do not immediately follow the head constituent, but this percentage drops to 11.1% for PPs that immediately follow the head. It seems, however, that increasing sister annotation beyond $s > 1$ only provide limited improvements.

In our second evaluation reported in Table 2, we

runs in polynomial time.

[6]To determine whether a given one-level tree is an auxiliary, we simply check the following properties: all its leaves but one (the "foot node") must be nodes attached to deleted subtrees (e.g., VP and CC in the figure), and the foot node (VP[9]) must have the same syntactic category as the root node.

[7]We relied on the SRI language modeling (SRILM) toolkit library for all smoothing experiments. We used the following order in our deleted interpolation of p_h: lexical head, head POS, ancestor annotation, and head category. For p_l and p_r, we removed first: lexical head, lexical head of the modifier, head POS, head POS of the modifier, sister annotation (L^i deleted before k_l^i), k_h, category of the head, category of the modifier. We experimented with different deletion interpolation orderings, and this ordering appears to work quite well in practice, and was used in all experiments reported in this paper.

assessed the usefulness of lexical and POS annotation (setting s and v to 0). In the table, we use M to denote any of the modifiers L_i or R_i, and c, t, w respectively represent syntactic constituent, POS, and lexical conditioning. While POS annotation is clearly advantageous compared to using only syntactic categories, adding lexical variables to the model also helps. As is shown in the table, it is especially important to know the lexical head of the modifier we are attempting to delete. The addition of w_m to conditioning variables provides an improvement of 1.3% (from 66.5% to 67.8%) on our optimal Ziff-Davis training corpus (ZD-6). Furthermore, bilexical head-modifier dependencies provide a relatively small improvement of .5% (from 69.8% to 70.3%) over the best model that does not incorporate the lexical head w_h. Note that lexical conditioning also helps in the case where the training data is relatively small (ZD-0), though differences are less significant, and bilexical dependencies actually hurt performance. In subsequent experiments, we experimented with different Markovizations and lexical dependency combination, and finally settled with a model ($s = 1$ and $v = 1$) incorporating all conditioning variables listed in the last line of Table 2. This final tuning was combined with human inspection of generated outputs, since certain modifications that positively impacted output quality seldom changed accuracies.

We finally took the best configuration selected above, and evaluated our model against the noisy-channel model of K&M on the 32 test sentences selected by them. We performed both automatic and human evaluation against the output produced by Knight and Marcu's original implementation of their noisy channel model (Table 3). In the former case, we also provide Simple String Accuracies (SSA).[8] For human evaluation, we hired six native-speaker judges who scored grammaticality and content (importance) with scores from 1 to 5, using instructions as described in K&M. Both types of evaluations favored our Markovized model against the noisy channel model.

Table 4 shows several outputs of our system

Vertical	Horizontal Order			
Order	$s=0$	$s=1$	$s=2$	$s=3$
$v=0$	63	67.1	67.2	67.2
$v=1$	63.9	67.6	**67.7**	67.7
$v=2$	65.7	66.6	66.9	66.9
$v=3$	65.2	66.8	67.1	67

Table 1: Markovizations accuracies on Ziff-Davis devel set.

Conditioning Variables		ZD-0	ZD-3	ZD-6
$M=c_m$	$H=c_h$	62.2	62.4	64.4
$M=(c_m,t_m)$	$H=c_h$	63.0	63.4	66.5
$M=(c_m,w_m)$	$H=c_h$	64.2	65.2	66.7
$M=(c_m,t_m,w_m)$	$H=c_h$	63.8	65.8	67.8
$M=(c_m,t_m,w_m)$	$H=(c_h,t_h)$	66.7	68.6	69.8
$M=(c_m,t_m,w_m)$	$H=(c_h,w_h)$	66.9	68.9	**70.3**
$M=(c_m,t_m,w_m)$	$H=(c_h,t_h,w_h)$	66.3	69.1	69.8

Table 2: Accuracies on Ziff-Davis devel set with different head-modifier annotations.

Models	Acc	SSA	Grammar	Content	Len(%)
NoisyC	61.3	14.6	4.37 ± 0.5	3.87 ± 1.2	70.4
Markov	67.9	31.7	4.68 ± 0.4	4.22 ± 0.4	62.7
Human	-	-	4.95 ± 0.1	4.43 ± 0.3	53.3

Table 3: Accuracies on Ziff-Davis test set.

(Markov) that significantly differed from the output of the noisy channel model (NoisyC), which confirms our finding that Markovized models can produce quite grammatical output. Our compression for the first sentence underlines one of the advantages of constituent-based classifiers, which have the ability of deleting a very long phrase (here, a PP) at once. The three next sentences display some advantages of our approach over the K&M model: here, the latter model performs deletion with too little lexico-syntactic information, and accidentally removes certain modifiers that are sometimes, but not always, good candidates for deletions (e.g., ADJP in Sentence 2, PP in sentences 3 and 4). On the other hand, our model keeps these constituent intact. Finally, the fifth and last example is one of the only three cases (among the 32 sentences) where our model produced a sentence we judged clearly ungrammatical. After inspection, we found that our parser assigned particularly errorful trees to those inputs, which may partially explain these ungrammatical outputs.

6 Related Work

A relatively large body of work addressed the problem of sentence compression. One successful recent approach (McDonald, 2006) combines a discriminative framework with a set of features that capture information similar to the K&M model. Mc-

[8]SSA is defined as: $SSA = 1 - (I + D + S)/R$. The numerator terms are respectively the number of inserts, deletes, and substitutions, and R is the length of the reference compression.

Input	*Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.*
NoisyC	Many debugging features, including user-defined points and variable-watching and message-watching windows, have been added.
Markov	Many debugging features have been added.
Human	Many debugging features have been added.
Input	*The chemical etching process used for glare protection is effective and will help if your office has the fluorescent-light overkill that 's typical in offices.*
NoisyC	The process used for glare protection is and will help if your office has the overkill
Markov	The chemical etching process used for glare protection is effective.
Human	Glare protection is effective.
Input	*The utilities will be bundled with Quickdex II in a $90 package called super quickdex, which is expected to ship in late summer.*
NoisyC	The utilities will be bundled
Markov	The utilities will be bundled with Quickdex II.
Human	The utilities will be bundled with Quickdex II.
Input	*The discounted package for the SparcServer 470 is priced at $89,900, down from the regular $107,795.*
NoisyC	The package for the 470 is priced
Markov	The discounted package for the SparcServer 470 is at $89,900.
Human	The SparcServer 470 is priced at $89,900, down from the regular $107,795.
Input	*Prices range from $5,000 for a microvax 2000 to $179,000 for the vax 8000 or higher series.*
NoisyC	Prices range from $5,000 for a 2000 to $179,000 for the vax 8000 or higher series.
Markov	Prices range from $5,000 for a microvax for the vax.
Human	Prices range from $5,000 to $179,000.

Table 4: Compressions of sample test sentences.

Donald's features include compression bigrams, as well as soft syntactic evidence extracted from parse trees and dependency trees. The strength of McDonald's approach partially stems from its robustness against redundant and noisy features, since each feature is weighted proportionally to its discriminative power, and his approach is thus hardly penalized by uninformative features. In contrast, our work puts much more emphasis on feature analysis than on efficient optimization, and relies on a statistical framework (maximum-likelihood estimates) that strives for careful feature selection and combination. It also describes and evaluates models incorporating syntactic evidence that is new to the sentence compression literature, such as head-modifier bilexical dependencies, and nth-order sister and vertical annotation. We think this work leads to a better understanding of what type of syntactic and lexical evidence makes sentence compression work. Furthermore, our work leaves the door open to uses of our factored model in a constituent-based or word-based discriminative framework, in which each elementary lexico-syntactic structure of this paper can be discriminatively weighted to directly optimize compression quality. Since McDonald's approach does not incorporate SCFG deletion rules, and conditions deletions on less lexico-syntactic context, we believe this will lead to levels of performance superior to both papers.

7 Conclusions

We presented a sentence compression system based on SCFG deletion rules, for which we defined a head-driven Markovization formulation. This Markovization enabled us to incorporate lexical conditioning variables into our models. We empirically evaluated different Markov structures, and obtained a best system that generates particularly grammatical sentences according to a human evaluation. Our sentence compression system is freely available for research and educational purposes.

Acknowledgments

We would like to thank Owen Rambow, Michael Collins, Julia Hirschberg, and Daniel Ellis for their helpful comments and suggestions.

References

A. Aho and J. Ullman. 1969. Syntax directed translations and the pushdown assembler. 3:37–56.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

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

B. Dorr, D. Zajic, and R. Schwartz. 2003. Hedge: A parse-and-trim approach to headline generation. In *Proc. of DUC*.

H. Jing. 2000. Sentence reduction for automatic text summarization. In *Proc. of NAACL*, pages 310–315.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

A. Joshi, L. Levy, and M. Takahashi. 1975. Tree adjunct grammar. *Journal of Computer and System Science*, 21(2).

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

K. Knight and D. Marcu. 2000. Statistics-based summarization — step one: Sentence compression. In *Proc. of AAAI*.

P. Lewis and R. Stearns. 1968. Syntax-directed transduction. In *Journal of the Association for Computing Machinery*, volume 15, pages 465–488.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proc. of EACL*.

J. Turner and E. Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proc. of ACL*.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.

Hybrid Models for Semantic Classification of Chinese Unknown Words

Xiaofei Lu
Department of Linguistics and Applied Language Studies
Pennsylvania State University
University Park, PA 16802, USA
xxl13@psu.edu

Abstract

This paper addresses the problem of classifying Chinese unknown words into fine-grained semantic categories defined in a Chinese thesaurus. We describe three novel knowledge-based models that capture the relationship between the semantic categories of an unknown word and those of its component characters in three different ways. We then combine two of the knowledge-based models with a corpus-based model which classifies unknown words using contextual information. Experiments show that the knowledge-based models outperform previous methods on the same task, but the use of contextual information does not further improve performance.

1 Introduction

Research on semantic annotation has focused primarily on word sense disambiguation (WSD), i.e., the task of determining the appropriate sense for each instance of a polysemous word out of a set of senses defined for the word in some lexicon. Much less work has been done on semantic classification of unknown words, i.e., words that are not listed in the lexicon. However, real texts typically contain a large number of unknown words. Successful classification of unknown words is not only useful for lexical acquisition, but also necessary for natural language processing (NLP) tasks that require semantic annotation.

This paper addresses the problem of classifying Chinese unknown words into fine-grained semantic categories defined in a Chinese thesaurus, *Cilin* (Mei et al., 1984). This thesaurus classifies over 70,000 words into 12 major categories, including human (A), concrete object (B), time and space (C), abstract object (D), attributes (E), actions (F), mental activities (G), activities (H), physical states (I), relations (J), auxiliaries (K), and honorifics (L). The 12 major categories are further divided into 94 medium categories, which in turn are subdivided into 1428 small categories. Each small category contains synonyms that are close in meaning. For example, under the major category *D*, the medium category *Dm* groups all words that refer to institutions, and the small category *Dm05* groups all words that refer to educational institutions, e.g., 学校 *xuéxiào* ‘school’. Unknown word classification involves a much larger search space than WSD. In classifying words into small categories in *Cilin*, the search space for a polysemous known word consists of all the categories the word belongs to, but that for an unknown word consists of all the 1428 small categories.

Research on WSD has concentrated on using contextual information, which may be limited for infrequent unknown words. On the other hand, Chinese characters carry semantic information that is useful for predicting the semantic properties of the words containing them. We present three novel knowledge-based models that capture the relationship between the semantic categories of an unknown word and those of its component characters in three different ways, and combine two of them with a corpus-based model that uses contextual information to classify unknown words. Experiments show that the combined knowledge-based model achieves an accuracy of 61.6% for classifying unknown words into small categories in *Cilin*, but the use of contextual information does not further improve performance.

The rest of the paper is organized as follows. Section 2 details the three novel knowledge-based models proposed for this task. Section 3 describes a corpus-based model. Section 4 reports the experiment results of the proposed models. Section 5 compares these results with previous results. Section 6 concludes the paper and points to avenues for further research.

2 Knowledge-based Models

This section describes three novel knowledge-based models for semantic classification of Chinese unknown words, including an overlapping-character model, a character-category association model, and a rule-based model. These models model the relationship between the semantic category of an unknown word and those of its component characters in three different ways.

2.1 The Baseline Model

The baseline model predicts the category of an unknown word by counting the number of overlapping characters between the unknown word and the member words in each category. As words in the same category are similar in meaning and the meaning of a Chinese word is generally the composition of the meanings of its characters, it is common for words in the same category to share one or more character. This model tests the hypothesis that speakers draw upon the repertoire of characters that relate to a concept when creating new words to realize it.

For each semantic category in *Cilin*, the set of unique characters in its member words are extracted, and the number of times each character occurs in word-initial, word-middle, and word-final positions is recorded. With this information, we develop two variants of the baseline model, which differ from each other in terms of whether it takes into consideration the positions in which the characters occur in words.

In variant 1, the score of a category is the sum of the number of occurrences of each character of the target word in the category, as in (1), where t_j denotes a category, w denotes the target word, c_i denotes the ith character in w, n is the length of w, and $f(c_i)$ is the frequency of c_i in t_j.

(1) $$Score(t_j, w) = \sum_{i=1}^{n} f(c_i)$$

In variant 2, the score of a category is the sum of the number of occurrences of each character of the unknown word in the category in its corresponding position, as in (2), where p_i denotes the position of c_i in w, which could be word-initial, word-middle, or word-final, and $f(c_i,p_i)$ denotes the frequency of c_i in position p_i in t_j.

(2) $$Score(t_j, w) = \sum_{i=1}^{n} f(c_i, p_i)$$

In each variant, the category with the maximum score for a target word is proposed as the category of the word.

2.2 Character-Category Associations

The relationship between the semantic category of an unknown word and those of its component characters can also be captured in a more sophisticated way using information-theoretical models. We use two statistical measures, mutual information and χ^2, to compute character-category associations and word-category associations. Chen (2004) used the χ^2 measure to compute character-character and word-word associations, but not word-category associations. We use word-category associations to directly predict the semantic categories of unknown words.

The mutual information and χ^2 measures are calculated as in (3) and (4), where $Asso(c,t_j)$ denotes the association between a character c and a semantic category t_j, and $P(X)$ and $f(X)$ denote the probability and frequency of X respectively.

(3) $$Asso_{MI}(c, t_j) = \log \frac{P(c, t_j)}{P(c)P(t_j)}$$

(4) $$Asso_{\chi^2}(c, t_j) = \frac{\alpha(c, t_j)}{\max_k \alpha(c, t_k)}$$

(5) $$\alpha(c, t_j) = \sqrt{\frac{[f(c, t_j)]^2}{f(c) + f(t_j)}}$$

Once the character-category associations are calculated, the association between a word w and a category t_j, $Asso(w,t_j)$, can be calculated as the sum of the weighted associations between each of the word's characters and the category, as in (6), where c_i denotes the ith character of w, $|w|$ denotes the length of w, and λ_i denotes the weight of $Asso(c_i,t_j)$. The λ's add up to 1. The weights are determined empirically based on the positions of the characters in the word.

(6) $$Asso(w, t_j) = \sum_{i=1}^{|w|} \lambda_i Asso(c_i, t_j)$$

As in variant 2 of the baseline model, the character-category association model can also be made sensitive to the positions in which the characters occur in the words. To this end, we first need to compute the position-sensitive associations between a category and a character in the word-initial, word-middle, and word-final positions separately. The position-sensitive association between an unknown word and a category can then be computed as the sum of the weighted position-sensitive associations between each of its characters and the category.

Once the word-category associations are computed, we can propose the highest ranked category or a ranked list of categories for each unknown word.

2.3 A Rule-Based Model

The third knowledge-based model uses linguistic rules to classify unknown words based on the syntactic and semantic categories of their component characters. Rule-based models have not been used for this task before. However, there are some regularities in the relationship between the semantic categories of unknown words and those of their component characters that can be captured in a more direct and effective way by linguistic rules than by statistical models.

A separate set of rules are developed for words of different lengths. Rules are initially developed based on knowledge about Chinese word formation, and are then refined by examining the development data. In general, the complete rule set takes a few hours to develop.

The rule in (7) is developed for bisyllabic unknown words. This rule proposes the common category of a bisyllabic word's two characters as its category. It is especially useful for words with a parallel structure, i.e., words whose two characters have the same meaning and syntactic category, e.g., 坍塌 *tāntā* 'collapse', where 坍 *tān* and 塌 *tā* both mean 'collapse' and share the category *Id05*. The thresholds for f_A and f_B are determined empirically and are both set to 1 if *AB* is a noun and to 0 and 3 respectively otherwise.

(7) For a bisyllabic word *AB*, if *A* and *B* share a category c^1, let f_A and f_B denote the number of times *A* and *B* occur in word-initial and word-final positions in *c* respectively. If f_A and f_B both surpass the predetermined thresholds, propose *c* for AB.

A number of rules are developed for trisyllabic words. While most rules in the model are general, the first rule in this set is rather specific, as it handles words with three specific prefixes, 大 *dà* 'big', 小 *xiǎo* 'little', and 老 *lǎo* 'old', which usually do not change the category of the root word. The other four rules again utilize the categories of the unknown word's component characters. The rules in (8b) and (8c) are similar to the rule in (7). The ones in (8d) and (8e) search for neighbor words with a similar structure as the target word. Eligible neighbors have a common morpheme with the target word in the same position and a second morpheme that shares a category with the second morpheme of the target word. For example, an eligible neighbor for 推销商 *tuīxiāo-shāng* 'sales-man' is 销售商 *xiāoshòu-shāng* 'distribut-or'. These two words share the morpheme 商 *shāng* 'businessman' in the word-final position, and the morphemes 推销 *tuīxiāo* 'to market' and 销售 *xiāoshòu* 'distribute' share the category *He03*. The rule in (8d) therefore applies in this case.

(8) For a trisyllabic word *ABC*:
   a. If *A* equals 大 *dà* 'big', 小 *xiǎo* 'little', or 老 *lǎo* 'old', propose the category of *AB* for *ABC* if *C* is the diminutive suffix 儿 *er* or the category of *BC* for *ABC* otherwise.
   b. If *A* and *BC* share a category *c*, propose *c* for *ABC*.
   c. If *AB* and *C* share a category *c*, propose *c* for *ABC*.
   d. If there is a word *XYC* such that *XY* and *AB* share a category, propose the category of *XYC* for *ABC*.
   e. If there is a word *XBC* such that *X* and *A* share a category, propose the category of *XBC* for *ABC*.

The rules for four-character words are given in (9). Like the rules in (8d) and (8e), these rules also search for neighbors of the target word.

(9) For a four-character word *ABCD*:
   a. If there is a word *XYZD/YZD* such that *XYZ/YZ* and *ABC* share a category, propose the category of *XYZ/YZ* for *ABCD*.
   b. If there is a word *ABCX* such that *X* and *D* share a category, propose the category of *ABCX* for *ABCD*.
   c. If there is a word *XYCD* such that *XY* and *AB* share a category, propose the category of *XYCD* for *ABCD*.
   d. If there is a word *XBCD/BCD*, propose the category of *XBCD/BCD* for *ABCD*.

3 A Corpus-Based Model

The knowledge-based models described above classify unknown words using information about the syntactic and semantic categories of their component characters. Another useful source of information is the context in which unknown words occur. While contextual information is the primary source of information used in WSD research and has been used for acquiring semantic lexicons and classifying unknown words in other languages (e.g., Roark and Charniak 1998; Ci-

[1] *A* and *B* may each belong to more than one category.

aramita 2003; Curran 2005), it has been used in only one previous study on semantic classification of Chinese unknown words (Chen and Lin, 2000). Part of the goal of this study is to investigate whether and how these two different sources of information can be combined to improve performance on semantic classification of Chinese unknown words.

To this end, we first use the knowledge-based models to propose a list of five candidate categories for the target word, then extract a generalized context for each category in *Cilin* from a corpus, and finally compute the similarity between the context of the target word and the generalized context of each of its candidate categories. Comparing the context of the target word with generalized contexts of categories instead of contexts of individual words alleviates the data-sparseness problem, as infrequent words have limited contextual information. Limiting the search space for each target word to the top five candidate categories reduces the computational cost that comes with the full search space.

3.1 Context Extraction and Representation

A generalized context for each semantic category is built from the contexts of its member words. This is done based on the assumption that as the words in the same category have the same or similar meaning, they tend to occur in similar contexts. In terms of context extraction and representation, we need to consider four factors.

Member Words The issue here is whether to include the contexts of polysemous member words in building the generalized context of a category. Including these contexts without discrimination introduces noise. To measure the effect of such noise, we build two versions of generalized context for each category, one using contexts of unambiguous member words only, and the other using contexts of all member words.

Context Words There are two issues in selecting words for context representation. First, words that contribute little information to the discrimination of meaning of other words, including conjunctions, numerals, auxiliaries, and non-Chinese sequences, are excluded. Second, to model the effect of frequency on the context words' contribution to meaning discrimination, we use two sets of context words: one consists of the 1000 most frequent words in the corpus; the other consists of all words in the corpus.

Window Size For WSD, both topical context and microcontext have been used (Ide and Véronis 1998). Topical context includes substantive words that co-occur with the target word within a larger window, whereas microcontext includes words in a small window around the target word. We experiment with topical context and microcontext with window sizes of 100 and 6 respectively (i.e., 50 and 3 words to the left and right of the target word respectively).

Context Representation We represent the context of a category as a vector $<w_1, w_2, ..., w_n>$, where n is the total number of context words, and w_i is the weight of the ith context word. To arrive at this representation, we first record the number of times each context word occurs within a specified window of each member word of a category in the corpus as a vector $<f_1, f_2, ..., f_n>$, where f_i is the number of times the ith context word co-occurs with a member word of the category. We then compute the weight of a context word w in context c, $W(w, c)$, using mutual information and t-test, which were reported by Weeds and Weir (2005) to perform the best on a pseudo-disambiguation task. These weight functions are computed as in (10) and (11), where N denotes the size of the corpus.

(10) $$W_{PMI}(w,c) = \log \frac{P(w,c)}{P(w)P(c)}$$

(11) $$W_t(w,c) = \frac{P(w,c) - P(w)P(c)}{\sqrt{P(w,c)/N}}$$

3.2 Contextual Similarity Measurement

We compute the similarity between the context vectors of the unknown word and its candidate categories using cosine. The cosine of two n-dimensional vectors $\vec{x}$ and $\vec{y}$, $cos(\vec{x}, \vec{y})$, is computed as in (12), where x_i and y_i denote the weight of the ith context word in $\vec{x}$ and $\vec{y}$.

(12) $$\cos(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

4 Results

4.1 Experiment Setup

The models are developed and tested using the Contemporary Chinese Corpus from Peking University (Yu et al. 2002) and the extended *Cilin* released by the Information Retrieval Lab at Harbin Institute of Technology. The corpus contains all the articles published in January, 1999 in *People's Daily*, a major newspaper in China. It contains over 1.12 million tokens and is word-segmented and POS-tagged. Table 1 summarizes the distribution of words in *Cilin*. Of the 76,029 words in *Cilin*, 35,151 are found in the Contemporary Chinese Corpus.

Length	Unambiguous	Polysemous	Total
1	2,674	2,068	4,742
2	39,057	5,403	44,460
3	15,112	752	15,864
4	9,397	942	10,338
≥5	590	34	624
Total	66,830	9,199	76,029

Table 1: Word distribution in the extended *Cilin*

We classify words into the third-level categories in the extended *Cilin*, which are equivalent to the small categories in the original *Cilin*. The development and test sets consist of 3,000 words each, which are randomly selected from the subset of words in *Cilin* that are two to four characters long, that have occurred in the Contemporary Chinese Corpus, and that are tagged as nouns, verbs, or adjectives in the corpus. The words in the development and test sets are also controlled for frequency, with 1/3 of them occurring 1-3 times, 3-6 times, and 7 or more times in the corpus respectively.

As Chen (2004) noted, excluding all the words in the development and test data in the testing stage worsens the data-sparseness problem for knowledge-based models, as some categories have few member words, and some characters appear in few words in some categories. To alleviate this problem, the remove-one method is used for testing the knowledge-based models. In other words, the models are re-trained for each test word using information about all the words in *Cilin except* the test word. The corpus-based model is trained once using the training data only, as the data-sparseness problem is alleviated by using generalized contexts of categories. Finally, if a word is polysemous, it is considered to have been correctly classified if the proposed category is one of its categories.

4.2 Results of the Baseline Model

Tables 2 and 3 summarize the results of the baseline model in terms of the accuracy of its best guess and best five guesses respectively.

The columns labeled "Non-filtered" report results where all categories are considered for each unknown word, and the ones labeled "POS-filtered" report results where only the categories that agree with the POS category of the unknown word are considered. In the latter case, if the target word is a noun, only the small categories under major categories A-D are considered; otherwise, only those under major categories E-L are considered. The results show that using POS information about the unknown word to filter categories improves performance. Variant 2 performs better when only the best guess is considered, indicating that it is useful to model the effect of position on a character's contribution to word meaning in this case. However, it is not helpful to be sensitive to character position when the best five guesses are considered.

Model	Non-filtered		POS-filtered	
variant	Dev	Test	Dev	Test
1	0.391	0.398	0.450	0.464
2	0.471	0.469	**0.514**	**0.517**

Table 2: Results of the baseline model: best guess

Model	Non-filtered		POS-filtered	
variant	Dev	Test	Dev	Test
1	0.757	0.760	**0.813**	**0.817**
2	0.764	0.762	0.809	0.805

Table 3: Results of the baseline model: best 5 guesses

4.3 Results of the Character-Category Association Model

In this model, only categories that agree with the POS category of the unknown word and that share at least one character with the unknown word are considered. These filtering steps significantly reduce the search space for this model.

We discussed three parameters of the model in Section 2.2, including the statistical measure, the sensitivity to character position in computing character-category associations, and the weights of the associations between categories and characters in different positions. In addition, the computation of the character-category associations can be sensitive or insensitive to the POS categories of the words containing the characters. In the POS-sensitive way, associations are computed among nouns (words in categories A-D) and non-nouns (words in categories E-L) separately, whereas in the POS-insensitive way, they are computed using all the words.

Tables 4 and 5 summarize the results of the character-category association model in terms of the accuracy of its best guess and best five guesses respectively. In all cases, the weights assigned to word-initial, word-middle, and word-final characters are 0.49, 0, and 0.51 respectively.

In terms of the best guess, the model achieves a best accuracy of 58.2%, a 6.5% improvement over the baseline result. The results show that χ^2 consistently performs better than mutual information, and computing position-sensitive character-category associations consistently improves performance. However, computing POS-sensitive associations gives mixed results.

In terms of the best five guesses, the model achieves a best accuracy of 83.8% on the test data, a 2.1% improvement over the best baseline result. Using χ^2 again achieves better results. However, in this case, the best results are achieved when the character-category associations are insensitive to both character position and the POS categories of words.

Sensitivity		Development		Test	
POS	Position	MI	χ^2	MI	χ^2
Yes	Yes	0.482	**0.586**	0.507	**0.582**
Yes	No	0.440	0.578	0.458	0.573
No	Yes	0.487	0.565	0.511	0.567
No	No	0.457	0.555	0.459	0.559

Table 4: Results of the character-category association model: best guess

Sensitivity		Development		Test	
POS	Position	MI	χ^2	MI	χ^2
Yes	Yes	0.735	0.805	0.720	0.810
Yes	No	0.743	0.828	0.754	0.821
No	Yes	0.702	0.813	0.718	0.812
No	No	0.735	**0.830**	0.746	**0.838**

Table 5: Results of the character-category association model: best 5 guesses

Word	Development			Test		
Len	R	P	F	R	P	F
2	0.159	0.796	0.265	0.158	0.772	0.262
3	0.368	0.838	0.511	0.351	0.830	0.493
4	0.582	0.852	0.692	0.540	0.900	0.675
All	0.218	**0.816**	0.344	0.216	**0.803**	0.340

Table 6: Results of the rule-based model: best guess

4.4 Results of the Rule-Based Model

Table 6 summarizes the results of the rule-based model in terms of recall, precision and F-score. The model returns multiple categories for some words, and it is considered to have correctly classified a word only when it returns a single, correct category for the word. Precision of the model is computed over all the cases where the model returns a single guess, and recall is computed over all cases. The model achieves an overall precision of 80.3% on the test data, much higher than the accuracy of the other two knowledge-based models. However, recall of the model is only 21.6%. The comparable results on the development and test sets indicate that the encoded rules are general. The model generally performs better on longer words than on shorter words.

4.5 Combining the Character-Category Association and Rule-Based Models

Given that the rule-based model achieves a higher precision but a lower recall than the character-category association model, the two models can be combined to improve the overall performance. In general, if the rule-based model returns one or more categories, these categories are first ranked among themselves by their associations with the unknown word. They are then followed by the other categories returned by the character-category association model. Tables 7 and 8 summarize the results of combining the two models.

Sensitivity		Development		Test	
POS	Position	MI	χ^2	MI	χ^2
Yes	Yes	0.561	**0.623**	0.572	**0.616**
Yes	No	0.536	0.622	0.542	0.615
No	Yes	0.562	0.610	0.575	0.608
No	No	0.530	0.601	0.532	0.606

Table 7: Results of combining the character-category association and rule-based models: best guess

Sensitivity		Development		Test	
POS	Position	MI	χ^2	MI	χ^2
Yes	Yes	0.834	0.846	0.845	0.843
Yes	No	0.791	**0.860**	0.801	0.851
No	Yes	0.760	0.848	0.742	0.845
No	No	0.773	0.859	0.782	**0.856**

Table 8: Results of combining the character-category association and rule-based models: best 5 guesses

In terms of the best guess, the combined model achieves an accuracy of 61.6%, a 3.4% improvement over the best result of the character-category association model alone. This is achieved using χ^2 with POS-sensitive and position-sensitive computation of character-category associations. In terms of the best five guesses, the model achieves an accuracy of 85.6%, a 1.8% improvement over the best result of the character-category association model alone.

To facilitate comparison with previous studies, the results of the combined model in terms of its best guess in classifying unknown words into major and medium categories are summarized in Table 9. As χ^2 consistently outperforms mutual information, results are reported for χ^2 only. With POS-sensitive and position-sensitive computation of character-category associations, the combined model achieves an accuracy of 83.0% and 69.9% for classifying unknown words into major and medium categories respectively.

Sensitivity		Development		Test	
POS	Position	Major	Med	Major	Med
Yes	Yes	**0.840**	**0.705**	**0.830**	**0.699**
Yes	No	0.831	0.698	0.828	0.698
No	Yes	0.832	0.692	0.825	0.692
No	No	0.821	0.687	0.821	0.689

Table 9: Results of the combined model for classifying unknown words into major and medium categories: best guess

4.6 Results of the Corpus-Based Model

The corpus-based model re-ranks the five highest ranked categories proposed by the combined knowledge-based model. Table 10 enumerates the parameters of the model and lists the labels used to denote the various settings in Table 11.

Parameter	Label	Setting	Label
Member words	MW	All members words	all
		Unambiguous members	un
Context words	CW	All words	all
		1000 most frequent	1000
Window size	WS	100	100
		6	6
Weight function	WF	Mutual information	mi
		t-test	t

Table 10: Parameter settings of the corpus-based model

Table 11 summarizes the results of 16 runs of the model with different parameter settings. The best accuracy on the test data is 37.1%, achieved in run 5 with the following parameter settings: using unambiguous member words for building contexts of categories, using all words in the corpus for context representation, using a window size of 100, and using mutual information as the weight function. As the combined knowledge-based model gives an accuracy of 85.6% for its best five guesses, the expected accuracy of a naive model that randomly picks a candidate for each word as its best guess is 17.1%. Compared with this baseline, the corpus-based model achieves a 13.0% improvement, but it performs much worse than the knowledge-based models.

Table 12 summarizes the accuracy of the top three runs of the model on words with different frequency in the corpus. Each of the three groups consists of 1,000 words that have occurred 1-2, 3-6, and 7 or more times in the corpus respectively. The model consistently performs better on words with higher frequency, suggesting that it may benefit from a larger corpus.

Run	Parameter Setting				Accuracy	
ID	MW	CW	WS	WF	Dev	Test
1	un	1000	100	mi	0.326	0.303
2	un	1000	100	t	0.317	0.288
3	un	1000	6	mi	0.304	0.301
4	un	1000	6	t	0.299	0.301
5	un	all	100	mi	0.359	**0.371**
6	un	all	100	t	0.292	0.296
7	un	all	6	mi	**0.370**	0.365
8	un	all	6	t	0.322	0.297
9	all	1000	100	mi	0.302	0.294
10	all	1000	100	t	0.314	0.304
11	all	1000	6	mi	0.313	0.314
12	all	1000	6	t	0.308	0.308
13	all	all	100	mi	0.336	0.333
14	all	all	100	t	0.287	0.300
15	all	all	6	mi	0.356	0.356
16	all	all	6	t	0.308	0.308

Table 11: Results of the corpus-based model

Run	Development			Test		
ID	1-2	3-6	≥7	1-2	3-6	≥7
5	0.331	0.360	0.385	0.323	0.389	0.402
7	0.323	0.363	0.423	0.335	0.357	0.402
15	0.328	0.346	0.395	0.334	0.355	0.379

Table 12: Results of the corpus-based model on words with different frequency

5 Related Work

The few previous studies on semantic classification of Chinese unknown word have primarily adopted knowledge-based models. Chen (2004) proposed a model that retrieves the word with the greatest association with the target word. This model is computationally more expensive than our character-category association model, as it entails computing associations between every character-category, category-character, character-character, and word-word pair. He reported an accuracy of 61.6% on bisyllabic V-V compounds. However, he included all the test words in training the model. If we also include the test words in computing character-category associations, the computationally cheaper model achieves an overall accuracy of 75.6%, with an accuracy of 75.1% on verbs.

Chen and Chen (2000) adopted similar exemplar-based models. Chen and Chen used a morphological analyzer to identify the head of the target word and the semantic categories of its modifier. They then retrieved examples with the same head as the target word. Finally, they computed the similarity between two words as the similarity between their modifiers, using the concept of information load (IC) of the least common ancestor (LCA) of the modifiers' semantic categories. They reported an accuracy of 81% for classifying 200 unknown nouns. Given the small test set of their study, it is hard to directly compare their results with ours.

Tseng used a morphological analyzer in the same way, but she also derived the morpho-syntactic relationship between the morphemes. She retrieved examples that share a morpheme with the target word in the same position and filtered those with a different morpho-syntactic relationship. Finally, she computed the similarity between two words as the similarity between their non-shared morphemes, using a similar concept of IC of the LCA of two categories. She classified unknown words into the 12 major categories only, and reported accuracies 65.8% on adjectives, 71.4% on nouns, and 52.8% on verbs. These results are not as good as the 83.0% overall accuracy our combined knowledge-based model achieved for classifying unknown words into major categories.

Chen and Lin (2000) is the only study that used contextual information for the same task. To generate candidate categories for a word, they looked up its translations in a Chinese-English dictionary and the synsets of the translations in WordNet, and mapped the synsets to the categories in *Cilin*. They used a corpus-based model similar to ours to rank the candidates. They reported an accuracy of 34.4%, which is close to the 37.1% accuracy of our corpus-based model, but lower than the 61.6% accuracy of our combined knowledge-based model. In addition, they could only classify the unknown words listed in the Chinese-English dictionary.

6 Conclusions

We presented three knowledge-based models and a corpus-based model for classifying Chinese unknown words into fine-grained categories in the Chinese thesaurus *Cilin*, a task important for lexical acquisition and NLP applications that require semantic annotation. The knowledge-based models use information about the categories of the unknown words' component characters, while the corpus-based model uses contextual information. By combining the character-category association and rule-based models, we achieved an accuracy of 61.6%. The corpus-based model did not improve performance.

Several avenues can be taken for further research. First, additional resources, such as bilingual dictionaries, morphological analyzers, parallel corpora, and larger corpora with richer linguistic annotation may prove useful for improving both the knowledge-based and corpus-based models. Second, we only explored one way to combine the knowledge-based and corpus-based models. Future work may explore alternative ways to combine these models to make better use of contextual information.

References

C.-J. Chen. 2004. Character-sense association and compounding template similarity: Automatic semantic classification of Chinese compounds. In *Proceedings of the 3rd SIGHAN Workshop on Chinese Language Processing*, pages 33–40.

M. Ciaramita and M. Johnson. 2003. Supersense tagging of unknown nouns in WordNet. In Proceedings of EMNLP-2003, pages 594-602.

K.-J. Chen and C.-J. Chen. 2000. Automatic semantic classification for Chinese unknown compound nouns. In *Proceedings of COLING-2000,* pages 173-179.

H.-H. Chen and C.-C. Lin. 2000. Sense-tagging Chinese corpus. In *Proceedings of the 2nd Chinese Language Processing Workshop*, pages 7-14.

J. Curran. 2005. Supersense tagging of unknown nouns using semantic similarity. In Proceedings of ACL-2006, pages 26-33.

N. Ide and J. Véronis. 1998. Introduction on the special issue on word sense disambiguation: The state of the art. *Computational Linguistics 24*(1):2–40.

J. Mei, Y. Zhu, Y. Gao, and H. Yin. (eds.) 1984. *Tongyici Cilin [A Thesaurus of Chinese Words]*. Commercial Press, Hong Kong.

B. Roark and E. Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of COLING/ACL-1998,* pages 1110-1116.

H. Tseng. 2003. Semantic classification of Chinese unknown words. In *Proceedings of ACL-2003 Student Research Workshop,* pages 72-79.

J. Weeds and D. Weir. 2005. Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics 31*(4):439–475.

S. Yu, H. Duan, X. Zhu, and B. Sun. 2002. The basic processing of Contemporary Chinese Corpus at Peking University. *Journal of Chinese Information Processing 16*(5):49–64.

Using Wikipedia for Automatic Word Sense Disambiguation

Rada Mihalcea
Department of Computer Science
University of North Texas
rada@cs.unt.edu

Abstract

This paper describes a method for generating sense-tagged data using Wikipedia as a source of sense annotations. Through word sense disambiguation experiments, we show that the Wikipedia-based sense annotations are reliable and can be used to construct accurate sense classifiers.

1 Introduction

Ambiguity is inherent to human language. In particular, word sense ambiguity is prevalent in all natural languages, with a large number of the words in any given language carrying more than one meaning. For instance, the English noun *plant* can mean *green plant* or *factory*; similarly the French word *feuille* can mean *leaf* or *paper*. The correct sense of an ambiguous word can be selected based on the context where it occurs, and correspondingly the problem of *word sense disambiguation* is defined as the task of automatically assigning the most appropriate meaning to a polysemous word within a given context.

Among the various knowledge-based (Lesk, 1986; Galley and McKeown, 2003; Navigli and Velardi, 2005) and data-driven (Yarowsky, 1995; Ng and Lee, 1996; Pedersen, 2001) word sense disambiguation methods that have been proposed to date, supervised systems have been constantly observed as leading to the highest performance. In these systems, the sense disambiguation problem is formulated as a supervised learning task, where each sense-tagged occurrence of a particular word is transformed into a feature vector which is then used in an automatic learning process. Despite their high performance, these supervised systems have an important drawback: their applicability is 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.

To address the sense-tagged data bottleneck problem, different methods have been proposed in the past, with various degrees of success. This includes the automatic generation of sense-tagged data using monosemous relatives (Leacock et al., 1998; Mihalcea and Moldovan, 1999; Agirre and Martinez, 2004), automatically bootstrapped disambiguation patterns (Yarowsky, 1995; Mihalcea, 2002), parallel texts as a way to point out word senses bearing different translations in a second language (Diab and Resnik, 2002; Ng et al., 2003; Diab, 2004), and the use of volunteer contributions over the Web (Chklovski and Mihalcea, 2002).

In this paper, we investigate a new approach for building sense tagged corpora using Wikipedia as a source of sense annotations. Starting with the hyperlinks available in Wikipedia, we show how we can generate sense annotated corpora that can be used for building accurate and robust sense classifiers. Through word sense disambiguation experiments performed on the Wikipedia-based sense tagged corpus generated for a subset of the SENSEVAL ambiguous words, we show that the Wikipedia annotations are reliable, and the quality of a sense tagging classifier built on this data set exceeds by a large margin the accuracy of an informed baseline that selects the most frequent word sense by default.

The paper is organized as follows. We first provide a brief overview of Wikipedia, and describe the view of Wikipedia as a sense tagged corpus. We then show how the hyperlinks defined in this resource can be used to derive sense annotated corpora, and we show how a word sense disambiguation system can be built on this dataset. We present the results obtained in the word sense disambiguation experiments, and conclude with a discussion of the results.

2 Wikipedia

Wikipedia is a free online encyclopedia, representing the outcome of a continuous collaborative effort of a large number of volunteer contributors. Virtually any Internet user can create or edit a Wikipedia webpage, and this "freedom of contribution" has a positive impact on both the quantity (fast-growing number of articles) and the quality (potential mistakes are quickly corrected within the collaborative environment) of this online resource. Wikipedia editions are available for more than 200 languages, with a number of entries varying from a few pages to more than one million articles per language.[1]

The basic entry in Wikipedia is an *article* (or *page*), which defines and describes an entity or an event, and consists of a hypertext document with hyperlinks to other pages within or outside Wikipedia. The role of the hyperlinks is to guide the reader to pages that provide additional information about the entities or events mentioned in an article.

Each article in Wikipedia is uniquely referenced by an identifier, which consists of one or more words separated by spaces or underscores, and occasionally a parenthetical explanation. For example, the article for *bar* with the meaning of *"counter for drinks"* has the unique identifier *bar (counter)*.[2]

The hyperlinks within Wikipedia are created using these unique identifiers, together with an *anchor text* that represents the surface form of the hyperlink. For instance, *"Henry Barnard, [[United States|American]] [[educationalist]], was born in [[Hartford, Connecticut]]"* is an example of a sentence in Wikipedia containing links to the articles *United States*, *educationalist*, and *Hartford, Connecticut*. If the surface form and the unique identifier of an article coincide, then the surface form can be turned directly into a hyperlink by placing double brackets around it (e.g. *[[educationalist]]*). Alternatively, if the surface form should be hyperlinked to an article with a different unique identifier, e.g. link the word *American* to the article on *United States*, then a piped link is used instead, as in *[[United States|American]]*.

One of the implications of the large number of contributors editing the Wikipedia articles is the occasional lack of consistency with respect to the unique identifier used for a certain entity. For instance, the concept of *circuit (electric)* is also referred to as *electronic circuit*, *integrated circuit*, *electric circuit*, and others. This has led to the so-called *redirect pages*, which consist of a redirection hyperlink from an alternative name (e.g. *integrated circuit*) to the article actually containing the description of the entity (e.g. *circuit (electric)*).

Finally, another structure that is particularly relevant to the work described in this paper is the *disambiguation page*. Disambiguation pages are specifically created for ambiguous entities, and consist of links to articles defining the different meanings of the entity. The unique identifier for a disambiguation page typically consists of the parenthetical explanation *(disambiguation)* attached to the name of the ambiguous entity, as in e.g. *circuit_(disambiguation)* which is the unique identifier for the disambiguation page of the entity *circuit*.

3 Wikipedia as a Sense Tagged Corpus

A large number of the concepts mentioned in Wikipedia are explicitly linked to their corresponding article through the use of links or piped links. Interestingly, these links can be regarded as *sense annotations* for the corresponding concepts, which is a property particularly valuable for entities that are ambiguous. In fact, it is precisely this observation that we rely on in order to generate sense tagged corpora starting with the Wikipedia annotations.

For example, ambiguous words such as e.g. *plant*, *bar*, or *chair* are linked to different Wikipedia articles depending on their meaning in the context where they occur. Note that the links are *manually* created by the Wikipedia users, which means that they are most of the time accurate and referencing

[1]In the experiments reported in this paper, we use a download from March 2006 of the English Wikipedia, with approximately 1 million articles, and more than 37 millions hyperlinks.

[2]The unique identifier is also used to form the article URL, e.g. http://en.wikipedia.org/wiki/Bar_(counter)

the correct article. The following represent five example sentences for the ambiguous word *bar*, with their corresponding Wikipedia annotations (links):

In 1834, Sumner was admitted to the **[[bar (law)|bar]]** at the age of twenty-three, and entered private practice in Boston.

It is danced in 3/4 time (like most waltzes), with the couple turning approx. 180 degrees every **[[bar (music)|bar]]**.

Vehicles of this type may contain expensive audio players, televisions, video players, and **[[bar (counter)|bar]]**s, often with refrigerators.

Jenga is a popular beer in the **[[bar (establishment)|bar]]**s of Thailand.

This is a disturbance on the water surface of a river or estuary, often cause by the presence of a **[[bar (landform)|bar]]** or dune on the riverbed.

To derive sense annotations for a given ambiguous word, we use the links extracted for all the hyperlinked Wikipedia occurrences of the given word, and map these annotations to word senses. For instance, for the *bar* example above, we extract five possible annotations: *bar (counter), bar (establishment), bar (landform), bar (law),* and *bar (music)*.

Although Wikipedia provides the so-called disambiguation pages that list the possible meanings of a given word, we decided to use instead the annotations collected directly from the Wikipedia links. This decision is motivated by two main reasons. First, a large number of the occurrences of ambiguous words are not linked to the articles mentioned by the disambiguation page, but to related concepts. This can happen when the annotation is performed using a concept that is similar, but not identical to the concept defined. For instance, the annotation for the word *bar* in the sentence *"The blues uses a rhythmic scheme of twelve 4/4 [[measure (music)|bars]]"* is *measure (music)*, which, although correct and directly related to the meaning of *bar (music)*, is not listed in the disambiguation page for *bar*.

Second, most likely due to the fact that Wikipedia is still in its incipient phase, there are several inconsistencies that make it difficult to use the disambiguation pages in an automatic system. For example, for the word *bar*, the Wikipedia page with the identifier *bar* is a disambiguation page, whereas for the word *paper*, the page with the identifier *paper* contains a description of the meaning of paper as *"material made of cellulose,"* and a different page *paper_(disambiguation)* is defined as a disambiguation page. Moreover, in other cases such as e.g. the entries for the word *organization*, no disambiguation page is defined; instead, the articles corresponding to different meanings of this word are connected by links labeled as "alternative meanings."

Therefore, rather than using the senses listed in a disambiguation page as the sense inventory for a given ambiguous word, we chose instead to collect all the annotations available for that word in the Wikipedia pages, and then map these labels to a widely used sense inventory, namely WordNet.[3]

3.1 Building Sense Tagged Corpora

Starting with a given ambiguous word, we derive a sense-tagged corpus following three main steps:

First, we extract all the paragraphs in Wikipedia that contain an occurrence of the ambiguous word as part of a link or a piped link. We select paragraphs based on the Wikipedia paragraph segmentation, which typically lists one paragraph per line.[4] To focus on the problem of word sense disambiguation, rather than named entity recognition, we explicitly avoid named entities by considering only those word occurrences that are spelled with a lower case. Although this simple heuristic will also eliminate examples where the word occurs at the beginning of a sentence (and therefore are spelled with an upper case), we decided nonetheless to not consider these examples so as to avoid any possible errors.

Next, we collect all the possible labels for the given ambiguous word by extracting the leftmost component of the links. For instance, in the piped link *[[musical_notation|bar]]*, the label *musical_notation* is extracted. In the case of simple links (e.g. *[[bar]]*), the word itself can also play the role of a valid label if the page it links to is not determined as a disambiguation page.

Finally, the labels are manually mapped to their corresponding WordNet sense, and a sense tagged

[3]Alternatively, the Wikipedia annotations could also play the role of a sense inventory, without the mapping to WordNet. We chose however to perform this mapping for the purpose of allowing evaluations using a widely used sense inventory.

[4]The average length of a paragraph is 80 words.

Word sense	Labels in Wikipedia	Wikipedia definition	WordNet definition
bar (establishment)	bar_(establishment), nightclub gay_club, pub	a retail establishment which serves alcoholic beverages	a room or establishment where alcoholic drinks are served over a counter
bar (counter)	bar_(counter)	the counter from which drinks are dispensed	a counter where you can obtain food or drink
bar (unit)	bar_(unit)	a scientific unit of pressure	a unit of pressure equal to a million dynes per square centimeter
bar (music)	bar_(music), measure_music musical_notation	a period of music	musical notation for a repeating pattern of musical beats
bar (law)	bar_association, bar_law law_society_of_upper_canada state_bar_of_california	the community of persons engaged in the practice of law	the body of individuals qualified to practice law in a particular jurisdiction
bar (landform)	bar_(landform)	a type of beach behind which lies a lagoon	a submerged (or partly submerged) ridge in a river or along a shore
bar (metal)	bar_metal, pole_(object)	-	a rigid piece of metal or wood
bar (sports)	gymnastics_uneven_bars, handle_bar	-	a horizontal rod that serves as a support for gymnasts as they perform exercises
bar (solid)	candy_bar, chocolate_bar	-	a block of solid substance

Table 1: Word senses for the word *bar*, based on annotation labels used in Wikipedia

corpus is created. This mapping process is very fast, as a relatively small number of labels is typically identified for a given word. For instance, for the dataset used in the experiments reported in Section 5, an average of 20 labels per word was extracted.

To ensure the correctness of this last step, for the experiments reported in this paper we used two human annotators who independently mapped the Wikipedia labels to their corresponding WordNet sense. In case of disagreement, a consensus was reached through adjudication by a third annotator. In a mapping agreement experiment performed on the dataset from Section 5, an inter-annotator agreement of 91.1% was observed with a kappa statistics of κ=87.1, indicating a high level of agreement.

3.2 An Example

As an example, consider the ambiguous word *bar*, with 1,217 examples extracted from Wikipedia where *bar* appeared as the rightmost component of a piped link or as a word in a simple link. Since the page with the identifier *bar* is a disambiguation page, all the examples containing the single link *[[bar]]* are removed, as the link does not remove the ambiguity. This process leaves us with 1,108 examples, from which 40 different labels are extracted. These labels are then manually mapped to nine senses in WordNet. Figure 1 shows the labels extracted from the Wikipedia annotations for the word *bar*, the corresponding WordNet definition, as well as the Wikipedia definition (when the sense was defined in the Wikipedia disambiguation page).

4 Word Sense Disambiguation

Provided a set of sense-annotated examples for a given ambiguous word, the task of a word sense disambiguation system is to automatically learn a disambiguation model that can predict the correct sense for a new, previously unseen occurrence of the word.

We use a word sense disambiguation system that integrates local and topical features within a machine learning framework, similar to several of the top-performing supervised word sense disambiguation systems participating in the recent SENSEVAL evaluations (http://www.senseval.org).

The disambiguation algorithm starts with a preprocessing step, where the text is tokenized and annotated with part-of-speech tags. Collocations are identified using a sliding window approach, where a collocation is defined as a sequence of words that forms a compound concept defined in WordNet.

Next, local and topical features are extracted from the context of the ambiguous word. Specifically, we use the current word and its part-of-speech, a local context of three words to the left and right of the ambiguous word, the parts-of-speech of the surrounding words, the verb and noun before and after the ambiguous words, and a global context implemented through sense-specific keywords determined as a list of at most five words occurring at least three times in the contexts defining a certain word sense.

This feature set is similar to the one used by (Ng and Lee, 1996), as well as by a number of state-of-the-art word sense disambiguation systems participating in the SENSEVAL-2 and SENSEVAL-3 evaluations. The features are integrated in a Naive Bayes classifier, which was selected mainly for its performance in previous work showing that it can lead to a state-of-the-art disambiguation system given the features we consider (Lee and Ng, 2002).

5 Experiments and Results

To evaluate the quality of the sense annotations generated using Wikipedia, we performed a word sense disambiguation experiment on a subset of the ambiguous words used during the SENSEVAL-2 and SENSEVAL-3 evaluations. Since the Wikipedia annotations are focused on nouns (associated with the entities typically defined by Wikipedia), the sense annotations we generate and the word sense disambiguation experiments are also focused on nouns.

Starting with the 49 ambiguous nouns used during the SENSEVAL-2 (29) and SENSEVAL-3 (20) evaluations, we generated sense tagged corpora following the process outlined in Section 3.1. We then removed all those words that have only one Wikipedia label (e.g. *detention*, which occurs 58 times, but appears as a single link *[[detention]]* in all the occurrences), or which have several labels that are all mapped to the same WordNet sense (e.g. *church*, which has 2,198 occurrences with several different labels such as *Roman church*, *Christian church*, *Catholic church*, which are all mapped to the meaning of *church, Christian church* as defined in WordNet). This resulted in a set of 30 words that have their Wikipedia annotations mapped to at least two senses according to the WordNet sense inventory.

Table 2 shows the disambiguation results using the word sense disambiguation system described in Section 4, using ten-fold cross-validation. For each word, the table also shows the number of senses, the total number of examples, and two baselines: a simple informed baseline that selects the most frequent sense by default,[5] and a more refined baseline that implements the corpus-based version of the Lesk algorithm (Kilgarriff and Rosenzweig, 2000).

[5]Note that this baseline assumes the availability of a sense tagged corpus in order to determine the most frequent sense of a word. The baseline is therefore "informed," as compared to a random, "uninformed" sense selection.

			baselines		word sense
word	#s	#ex	MFS	LeskC	disambig.
argument	2	114	70.17%	73.63%	**89.47%**
arm	3	291	61.85%	69.31%	**84.87%**
atmosphere	3	773	54.33%	56.62%	**71.66%**
bank	3	1074	**97.20%**	**97.20%**	**97.20%**
bar	10	1108	47.38%	68.09%	**83.12%**
chair	3	194	67.57%	65.78%	**80.92%**
channel	5	366	51.09%	52.50%	**71.85%**
circuit	4	327	85.32%	85.62%	**87.15%**
degree	7	849	58.77%	73.05%	**85.98%**
difference	2	24	**75.00%**	**75.00%**	**75.00%**
disc	3	73	52.05%	52.05%	**71.23%**
dyke	2	76	77.63%	82.00%	**89.47%**
fatigue	3	123	66.66%	70.00%	**93.22%**
grip	3	34	44.11%	**77.00%**	70.58%
image	2	84	69.04%	74.50%	**80.28%**
material	3	223	**95.51%**	**95.51%**	**95.51%**
mouth	2	409	94.00%	94.00%	**95.35%**
nature	2	392	**98.72%**	**98.72%**	98.21%
paper	5	895	**96.98%**	**96.98%**	**96.98%**
party	3	764	68.06%	68.28%	**75.91%**
performance	2	271	**95.20%**	**95.20%**	**95.20%**
plan	3	83	77.10%	81.00%	**81.92%**
post	5	33	54.54%	**62.50%**	51.51%
restraint	2	9	**77.77%**	**77.77%**	**77.77%**
sense	2	183	**95.10%**	**95.10%**	**95.10%**
shelter	2	17	**94.11%**	**94.11%**	**94.11%**
sort	2	11	81.81%	**90.90%**	**90.90%**
source	3	78	55.12%	81.00%	**92.30%**
spade	3	46	60.86%	**81.50%**	80.43%
stress	3	565	53.27%	54.28%	**86.37%**
AVERAGE	3.31	316	72.58%	78.02%	**84.65%**

Table 2: Word sense disambiguation results, including two baselines (MFS = most frequent sense; LeskC = Lesk-corpus) and the word sense disambiguation system. Number of senses (#s) and number of examples (#ex) are also indicated.

6 Discussion

Overall, the Wikipedia-based sense annotations were found reliable, leading to accurate sense classifiers with an average relative error rate reduction of 44% compared to the most frequent sense baseline, and 30% compared to the Lesk-corpus baseline.

There were a few exceptions to this general trend. For instance, for some of the words for which only a small number of examples could be collected from Wikipedia, e.g. *restraint* or *shelter*, no accuracy improvement was observed compared to the most frequent sense baseline. Similarly, several words in the

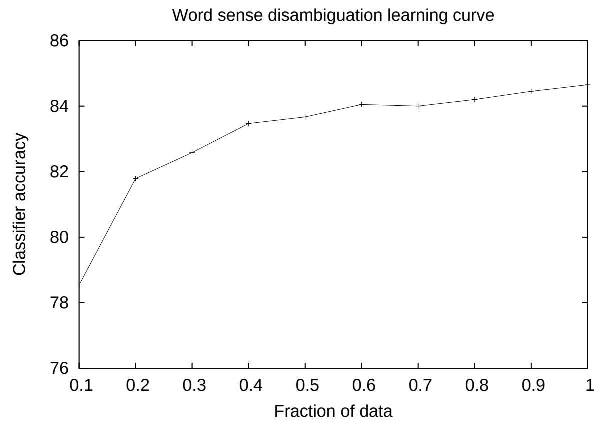


Figure 1: Learning curve on the Wikipedia data set.

data set have highly skewed sense distributions, such as e.g. *bank*, which has a total number of 1,074 examples out of which 1,044 examples pertain to the meaning of *financial institution*, or the word *material* with 213 out of 223 examples annotated with the meaning of *substance*.

One aspect that is particularly relevant for any supervised system is the learning rate with respect to the amount of available data. To determine the learning curve, we measured the disambiguation accuracy under the assumption that only a fraction of the data were available. We ran ten fold cross-validation experiments using 10%, 20%, ..., 100% of the data, and averaged the results over all the words in the data set. The resulting learning curve is plotted in Figure 1. Overall, the curve indicates a continuously growing accuracy with increasingly larger amounts of data. Although the learning pace slows down after a certain number of examples (about 50% of the data currently available), the general trend of the curve seems to indicate that more data is likely to lead to increased accuracy. Given that Wikipedia is growing at a fast pace, the curve suggests that the accuracy of the word sense classifiers built on this data is likely to increase for future versions of Wikipedia.

Another aspect we were interested in was the correlation in terms of sense coverage with respect to other sense annotated data currently available. For the set of 30 nouns in our data set, we collected all the word senses that were defined in either the Wikipedia-based sense-tagged corpus or in the SENSEVAL corpus. We then determined the percentage covered by each sense with respect to the entire data set available for a given ambiguous word. For instance, the noun *chair* appears in Wikipedia with senses #1 (68.0%), #2 (31.9%), and #4(0.1%), and in SENSEVAL with senses #1 (87.7%), #2 (6.3%), and #3 (6.0%). The senses that do not appear are indicated with a 0% coverage. The correlation is then measured between the relative sense frequencies of all the words in our dataset, as observed in the two corpora. Using the Pearson (r) correlation factor, we found an overall correlation of $r = 0.51$ between the sense distributions in the Wikipedia corpus and the SENSEVAL corpus, which indicates a medium correlation. This correlation is much lower than the one observed between the sense distributions in the training data and in the test data in the SENSEVAL corpus, which was measured at a high $r = 0.95$. This suggests that the sense coverage in Wikipedia follows a different distribution than in SENSEVAL, mainly reflecting the difference between the genres of the two corpora: an online collection of encyclopedic pages as available from Wikipedia, versus the manually balanced British National Corpus used in SENSEVAL. It also suggests that using the Wikipedia-based sense tagged corpus to disambiguate words in the SENSEVAL data or viceversa would require a change in the distribution of senses as previously done in (Agirre and Martinez, 2004).

			baselines		word sense
Dataset	#s	#ex	MFS	LeskC	disambig.
SENSEVAL	4.60	226	51.53%	58.33%	68.13%
WIKIPEDIA	3.31	316	72.58%	78.02%	84.65%

Table 3: Average number of senses and examples, most frequent sense and Lesk-corpus baselines, and word sense disambiguation performance on the SENSEVAL and WIKIPEDIA datasets.

Table 3 shows the characteristics of the SENSEVAL and the WIKIPEDIA datasets for the nouns listed in Table 2. The table also shows the most frequent sense baseline, the Lesk-corpus baseline, as well as the accuracy figures obtained on each dataset using the word sense disambiguation system described in Section 4.[6]

[6]As a side note, the accuracy obtained by our system on the SENSEVAL data is comparable to that of the best participating systems. Using the output of the best systems: the JHU$_R$ system on the SENSEVAL-2 words, and the HLTS$_3$ system on the

Overall the sense distinctions identified in Wikipedia are fewer and typically coarser than those found in WordNet. As shown in Table 3, for the set of ambiguous words listed in Table 2, an average of 4.6 senses were used in the SENSEVAL annotations, as compared to about 3.3 senses per word found in Wikipedia. This is partly due to a different sense coverage and distribution in the Wikipedia data set (e.g. the meaning of *ambiance* for the ambiguous word *atmosphere* does not appear at all in the Wikipedia corpus, although it has the highest frequency in the SENSEVAL data), and partly due to the coarser sense distinctions made in Wikipedia (e.g. Wikipedia does not make the distinction between the act of grasping and the actual hold for the noun *grip*, and occurrences of both of these meanings are annotated with the label *grip_(handle)*).

There are also cases when Wikipedia makes different or finer sense distinctions than WordNet. For instance, there are several Wikipedia annotations for *image* as *copy*, but this meaning is not even defined in WordNet. Similarly, Wikipedia makes the distinction between *dance performance* and *theatre performance*, but both these meanings are listed under one single entry in WordNet (*performance* as *public presentation*). However, since at this stage we are mapping the Wikipedia annotations to WordNet, these differences in sense granularity are diminished.

7 Related Work

In word sense disambiguation, the line of work most closely related to ours consists of methods trying to address the sense-tagged data bottleneck problem.

A first set of methods consists of algorithms that generate sense annotated data using words semantically related to a given ambiguous word (Leacock et al., 1998; Mihalcea and Moldovan, 1999; Agirre and Martinez, 2004). Related non-ambiguous words, such as monosemous words or phrases from dictionary definitions, are used to automatically collect examples from the Web. These examples are then turned into sense-tagged data by replacing the non-ambiguous words with their ambiguous equivalents.

Another approach proposed in the past is based on the idea that an ambiguous word tends to have different translations in a second language (Resnik and Yarowsky, 1999). Starting with a collection of parallel texts, sense annotations were generated either for one word at a time (Ng et al., 2003; Diab, 2004), or for all words in unrestricted text (Diab and Resnik, 2002), and in both cases the systems trained on these data were found to be competitive with other word sense disambiguation systems.

The lack of sense-tagged corpora can also be circumvented using bootstrapping algorithms, which start with a few annotated seeds and iteratively generate a large set of disambiguation patterns. This method, initially proposed by (Yarowsky, 1995), was successfully evaluated in the context of the SENSEVAL framework (Mihalcea, 2002).

Finally, in an effort related to the Wikipedia collection process, (Chklovski and Mihalcea, 2002) have implemented the Open Mind Word Expert system for collecting sense annotations from volunteer contributors over the Web. The data generated using this method was then used by the systems participating in several of the SENSEVAL-3 tasks.

Notably, the method we propose has several advantages over these previous methods. First, our method relies exclusively on monolingual data, thus avoiding the possible constraints imposed by methods that require parallel texts, which may be difficult to find. Second, the Wikipedia-based annotations follow a natural Zipfian sense distribution, unlike the equal distributions typically obtained with the methods that rely on the use of monosemous relatives or bootstrapping methods. Finally, the grow pace of Wikipedia is much faster than other more task-focused and possibly less-engaging activities such as Open Mind Word Expert, and therefore has the potential to lead to significantly higher coverage.

With respect to the use of Wikipedia as a resource for natural language processing tasks, the work that is most closely related to ours is perhaps the name entity disambiguation algorithm proposed in (Bunescu and Pasca, 2006), where an SVM kernel is trained on the entries found in Wikipedia for ambiguous named entities. Other language processing tasks with recently proposed solutions relying on Wikipedia are co-reference resolution using Wikipedia-based measures of word similarity (Strube and Ponzetto, 2006), enhanced text classification using encyclopedic knowledge (Gabrilovich

SENSEVAL-3 words, an average accuracy of 71.31% was measured (the output of the systems participating in SENSEVAL is publicly available from http://www.senseval.org).

and Markovitch, 2006), and the construction of comparable corpora using the multilingual editions of Wikipedia (Adafre and de Rijke, 2006).

8 Conclusions

In this paper, we described an approach for using Wikipedia as a source of sense annotations for word sense disambiguation. Starting with the hyperlinks available in Wikipedia, we showed how we can generate a sense annotated corpus that can be used to train accurate sense classifiers. Through experiments performed on a subset of the SENSEVAL words, we showed that the Wikipedia sense annotations can be used to build a word sense disambiguation system leading to a relative error rate reduction of 30–44% as compared to simpler baselines.

Despite some limitations inherent to this approach (definitions and annotations in Wikipedia are available almost exclusively for nouns, word and sense distributions are sometime skewed, the annotation labels are occasionally inconsistent), these limitations are overcome by the clear advantage that comes with the use of Wikipedia: large sense tagged data for a large number of words at virtually no cost.

We believe that this approach is particularly promising for two main reasons. First, the size of Wikipedia is growing at a steady pace, which consequently means that the size of the sense tagged corpora that can be generated based on this resource is also continuously growing. While techniques for supervised word sense disambiguation have been repeatedly criticized in the past for their limited coverage, mainly due to the associated sense-tagged data bottleneck, Wikipedia seems a promising resource that could provide the much needed solution for this problem. Second, Wikipedia editions are available for many languages (currently about 200), which means that this method can be used to generate sense tagged corpora and build accurate word sense classifiers for a large number of languages.

References

S. F. Adafre and M. de Rijke. 2006. Finding similar sentences across multiple languages in wikipedia. In *Proceedings of the EACL Workshop on New Text*, Trento, Italy.

E. Agirre and D. Martinez. 2004. Unsupervised word sense disambiguation based on automatically retrieved examples: The importance of bias. In *Proceedings of EMNLP 2004*, Barcelona, Spain, July.

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL 2006*, Trento, Italy.

T. Chklovski and R. Mihalcea. 2002. Building a sense tagged corpus with Open Mind Word Expert. In *Proceedings of the ACL 2002 Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions"*, Philadelphia, July.

M. Diab and P. Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of ACL 2002*, Philadelphia.

M. Diab. 2004. Relieving the data acquisition bottleneck in word sense disambiguation. In *Proceedings of ACL 2004*, Barcelona, Spain.

E. Gabrilovich and S. Markovitch. 2006. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of AAAI 2006*, Boston.

M. Galley and K. McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of IJCAI 2003*, Acapulco, Mexico.

A. Kilgarriff and R. Rosenzweig. 2000. Framework and results for English SENSEVAL. *Computers and the Humanities*, 34:15–48.

C. Leacock, M. Chodorow, and G.A. Miller. 1998. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24(1):147–165.

Y.K. Lee and H.T. Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP 2002*, Philadelphia.

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, June.

R. Mihalcea and D.I. Moldovan. 1999. An automatic method for generating sense tagged corpora. In *Proceedings of AAAI 1999*, Orlando.

R. Mihalcea. 2002. Bootstrapping large sense tagged corpora. In *Proceedings of LREC 2002*, Canary Islands, Spain.

R. Navigli and P. Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27.

H.T. Ng and H.B. Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An examplar-based approach. In *Proceedings of ACL 1996*, New Mexico.

H.T. Ng, B. Wang, and Y.S. Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of ACL 2003*, Sapporo, Japan.

T. Pedersen. 2001. A decision tree of bigrams is an accurate predictor of word sense. In *Proceedings of NAACL 2001*, Pittsburgh.

P. Resnik and D. Yarowsky. 1999. Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5(2):113–134.

M. Strube and S. P. Ponzetto. 2006. Wikirelate! computing semantic relatedeness using Wikipedia. In *Proceedings of AAAI 2006*, Boston.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL 1995*, Cambridge.

Data-Driven Graph Construction for Semi-Supervised Graph-Based Learning in NLP

Andrei Alexandrescu
Dept. of Computer Science and Engineering
University of Washington
Seattle, WA, 98195
andrei@cs.washington.edu

Katrin Kirchhoff
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195
katrin@ee.washington.edu

Abstract

Graph-based semi-supervised learning has recently emerged as a promising approach to data-sparse learning problems in natural language processing. All graph-based algorithms rely on a graph that jointly represents labeled and unlabeled data points. The problem of how to best construct this graph remains largely unsolved. In this paper we introduce a data-driven method that optimizes the representation of the initial feature space for graph construction by means of a supervised classifier. We apply this technique in the framework of label propagation and evaluate it on two different classification tasks, a multi-class lexicon acquisition task and a word sense disambiguation task. Significant improvements are demonstrated over both label propagation using conventional graph construction and state-of-the-art supervised classifiers.

1 Introduction

Natural Language Processing (NLP) applications benefit from the availability of large amounts of annotated data. However, such data is often scarce, particularly for non-mainstream languages. Semi-supervised learning addresses this problem by combining large amounts of unlabeled data with a small set of labeled data in order to learn a classification function. One class of semi-supervised learning algorithms that has recently attracted increased interest is graph-based learning. Graph-based techniques represent labeled and unlabeled data points as nodes in a graph with weighted edges encoding the similarity of pairs of samples. Various techniques are then available for transferring class labels from the labeled to the unlabeled data points. These approaches have shown good performance in cases where the data is characterized by an underlying manifold structure and samples are judged to be similar by local similarity measures. However, the question of how to best construct the graph forming the basis of the learning procedure is still an under-investigated research problem. NLP learning tasks present additional problems since they often rely on discrete or heterogeneous feature spaces for which standard similarity measures (such as Euclidean or cosine distance) are suboptimal.

We propose a two-pass data-driven technique for graph construction in the framework of label propagation (Zhu, 2005). First, we use a supervised classifier trained on the labeled subset to transform the initial feature space (consisting of e.g. lexical, contextual, or syntactic features) into a continuous representation in the form of soft label predictions. This representation is then used as a basis for measuring similarity among samples that determines the structure of the graph used for the second, semi-supervised learning step. It is important to note that, rather than simply cascading the supervised and the semi-supervised learner, we optimize the combination with respect to the properties required of the graph. We present several techniques for such optimization, including regularization of the first-pass classifier, biasing by class priors, and linear combination of classifier predictions with known features.

The proposed approach is evaluated on a lexicon learning task using the Wall Street Journal (WSJ) corpus, and on the SENSEVAL-3 word sense disambiguation task. In both cases our technique significantly outperforms our baseline systems (label propagation using standard graph construction and discriminatively trained supervised classifiers).

2 Background

Several graph-based learning techniques have recently been developed and applied to NLP problems: minimum cuts (Pang and Lee, 2004), random walks (Mihalcea, 2005; Otterbacher et al., 2005), graph matching (Haghighi et al., 2005), and label propagation (Niu et al., 2005). Here we focus on label propagation as a learning technique.

2.1 Label propagation

The basic label propagation (LP) algorithm (Zhu and Ghahramani, 2002; Zhu, 2005) has as inputs:

- a labeled set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i are samples (feature vectors) and $y_i \in \{1, 2, \ldots, C\}$ are their corresponding labels;
- an unlabeled set $\{x_{n+1}, \ldots, x_N\}$;
- a distance measure $d(i, j)$ $i, j \in \{1, \ldots N\}$ defined on the feature space.

The goal is to infer the labels $\{y_{n+1}, \ldots, y_N\}$ for the unlabeled set. The algorithm represents all N data points as vertices in an undirected graph with weighted edges. Initially, only the known data vertices are labeled. The edge linking vertices i and j has weight:

$$w_{ij} = \exp\left(-\frac{d(i,j)^2}{\alpha^2}\right) \quad (1)$$

where α is a hyperparameter that needs to be empirically chosen or learned separately. w_{ij} indicates the label affinity of vertices: the larger w_{ij} is, the more likely it is that i and j have the same label. The LP algorithm constructs a row-normalized $N \times N$ transition probability matrix P as follows:

$$P_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_{k=1}^{N} w_{ik}} \quad (2)$$

The algorithm probabilistically pushes labels from the labeled nodes to the unlabeled nodes. To do so, it defines the $n \times C$ hard labels matrix Y and the $N \times C$ soft labels matrix f, whose first n rows are identical to Y. The hard labels matrix Y is invariant through the algorithm and is initialized with probability 1 for the known label and 0 for all other labels:

$$Y_{ic} = \delta(y_i, C) \quad (3)$$

where δ is Kronecker's delta function. The algorithm iterates as follows:

```
1. $f' \leftarrow P \times f$
2. $f'_{[rows\ 1\ to\ n]} \leftarrow Y$
3. If $f' \cong f$, stop
4. $f \leftarrow f'$
5. Repeat from step 1
```

In each iteration, step 2 fixes the known labels, which might otherwise be overriden by propagated labels. The resulting labels for each feature x_i, where $i \in \{n+1, \ldots, N\}$, are:

$$l_i = \arg\max_{j=1,\ldots,C} f_{ij} \quad (4)$$

It is important that the distance measure is locally accurate, i.e. nodes connected by an edge with a high weight should have the same label. The global distance is less relevant since label information will be propagated from labeled points through the entire space. This is why LP works well with a local distance measure that might be unsuitable as a global distance measure.

Applications of LP include handwriting recognition (Zhu and Ghahramani, 2002), image classification (Balcan et al., 2005) and retrieval (Qin et al., 2005), and protein classification (Weston et al., 2003). In NLP, label propagation has been used for word sense disambiguation (Niu et al., 2005), document classification (Zhu, 2005), sentiment analysis (Goldberg and Zhu, 2006), and relation extraction (Chen et al., 2006).

2.2 Graph construction

One of the main problems in LP, as well as other graph-based learning techniques, is how to best construct the graph. Currently, graph construction "is more of an art than science" (Zhu, 2005). Typically, edge weights are derived from a simple Euclidean or cosine distance measure, regardless of the nature of the underlying features. Edges are then established either by connecting all nodes, by applying a single global threshold to the edge weights, or by connecting each node to its k nearest neighbors according to the edge weights. This procedure is often suboptimal: Euclidean distance relies on a model of normally distributed i.i.d. random variables; cosine distance likewise assumes that the different feature vector dimensions are uncorrelated. However, many applications, particularly in NLP, rely on feature spaces with correlated dimensions. Moreover, features may have different ranges and different types (e.g. continuous, binary, multi-valued), which entails the need for normalization, binning, or scaling. Finally, common distance measures do not take advantage of domain knowledge that might be available.

Some attempts have been made at improving the standard method of graph construction. For instance, in a face identification task (Balcan et al., 2005), domain knowledge was used to identify three different edge sets based on time, color and face features, associating a different hyperparameter with each. The resulting graph was then created by superposing edge sets. Zhu (Zhu, 2005, Ch. 7) describes graph construction using separate α hyperparameters for each feature dimension, and presents a data-driven way (evidence maximization) for learning the values of the parameters.

3 Data-driven graph construction

Unlike previous work, we propose to optimize the feature representation used for graph construction by learning it with a first-pass supervised classifier. Under this approach, similarity of samples is defined as similarity of the output values produced by a classifier applied to the original feature representation of the samples. This idea bears similarity to classifier cascading (Alpaydin and Kaynak, 1998), where classifiers are trained around a rule-exceptions paradigm; however, in our case, the classifiers work together, the first acting as a jointly optimized feature mapping function for the second.

1. Train a first-pass supervised classifier that outputs soft label predictions Z_i for all samples $i \in \{1, \dots N\}$, e.g. a posterior probability distribution over target labels: $Z_i = \langle p_{i1}, p_{i2}, \dots, p_{iC} \rangle$;
2. Apply postprocessing to Z_i if needed.
3. Use vectors Z_i and an appropriately chosen distance measure to construct a graph for LP.
4. Perform label propagation over the constructed graph to find the labeling of the test samples.

The advantages of this procedure are:

• *Uniform range and type of features*: The output from a first-pass classifier can produce well-defined features, e.g. posterior probability distributions. This eliminates the problem of input features of different ranges and types (e.g. binary vs. multi-valued, continuous vs. categorical attributes) which are often used in combination.

• *Feature postprocessing*: The transformation of features into a different space also opens up possibilities for postprocessing (e.g. probability distribution warping) depending on the requirements of the second-pass learner. In addition, different distance functions (e.g. those defined on probability spaces) can be used, which avoids violating assumptions made by metrics such as Euclidean and cosine distance.

• *Optimizing class separation:* The learned representation of labeled training samples might reveal better clusters in the data than the original representation: a discriminatively-trained first pass classifier will attempt to maximize the separation of samples belonging to different classes. Moreover, the first-pass classifier may learn a feature transformation that suppresses noise in the original input space.

Difficulties with the proposed approach might arise when the first-pass classifier yields confident but wrong predictions, especially for outlier samples in the original space. For this reason, the first-pass classifier and the graph-based learner should not simply be concatenated without modification, but the first classifier should be optimized with respect to the requirements of the second. In our case, the choice of first-pass classifier and joint optimization techniques are determined by the particular learning task and are detailed below.

4 Tasks

4.1 Lexicon acquisition task

Our first task is a part-of-speech (POS) lexicon acquisition task, i.e. the labels to be predicted are the sets of POS tags associated with each word in a lexicon. Note that this is *not* a tagging task: we are not attempting to identify the correct POS of each word in running text. Rather, for each word in the vocabulary, we attempt to infer the set of *possible* POS tags. Our choice of this task is motivated by our long-term goal of applying this technique to lexicon acquisition for resource-poor languages: POS lexicons are one of the most basic language resources, which enable subsequent training of taggers, chunkers, etc. We assume that a small set of words can be reliably annotated, and that POS-sets for the remaining words can be inferred by semi-supervised learning. Rather than choosing a genuinely resource-poor language for this task, we use the English Wall Street Journal (WSJ) corpus and artificially limit the size of the labeled set. This is because the WSJ corpus is widely obtainable and allows easy replication of our experiments.

We use sections 0-18 of the Wall Street Journal corpus ($N = 44,492$). Words have between 1 and 4 POS tags, with an average of 1.1 per word. The number of POS tags is 36, and we treat every POS combination as a unique class, resulting in $C = 158$ distinct labels. We use three different randomly selected training sets of various sizes: 5000, 10000, and 15000 words, representing about 11%, 22%, and 34% of the entire data set respectively; the rest of the data was used for testing. In order to avoid experimental bias, we run all experiments on five different randomly chosen labeled subsets and report averages and standard deviations. Due to the random sampling of the data it is possible that some labels never occur in the training set or only occur once. We train our classifiers only on those labels that occur at least twice, which results in 60-63 classes. Labels not present in the training set will therefore not be hypothesized and are guaranteed to be errors. We delete samples with unknown labels from our unlabeled set since their percentage is less than 0.5% on average.

We use the following features to represent samples:

- Integer: the three-letter suffix of the word;
- Integer: The four-letter suffix of the word;
- Integer $\times$ 4: The indices of the four most frequent words that immediately precede the word in the WSJ text;
- Boolean: word contains capital letters;
- Boolean: word consists only of capital letters;
- Boolean: word contains digits;
- Boolean: word contains a hyphen;
- Boolean: word contains other special characters (e.g. "&").

We have also experimented with shorter suffixes and with prefixes but those features tended to degrade performance.

4.2 SENSEVAL-3 word sense disambiguation task

The second task is word sense disambiguation using the SENSEVAL-3 corpus (Mihalcea et al., 2004), to enable a comparison of our method with previously published results. The goal is to disambiguate the different senses of each of 57 words given the sentences within which they occur. There are 7860 samples for training and 3944 for testing. In line with existing work (Lee and Ng, 2002; Niu et al., 2005), we use the following features:

- Integer $\times$ 7: seven features consisting of the POS of the previous three words, the POS of the next three words, and the POS of the word itself. We used the MXPOST tagger (Ratnaparkhi, 1996) for POS annotation.
- Integer $\times\langle variable\ length\rangle$: a bag of all words in the surrounding context.
- Integer $\times$ 15: Local collocations C_{ij} (i, j are the bounds of the collocation window)—word combinations from the context of the word to disambiguate. In addition to the 11 collocations used in similar work (Lee and Ng, 2002), we also used $C_{-3,1}$, $C_{-3,2}$, $C_{-2,3}$, $C_{-1,3}$.

Note that syntactic features, which have been used in some previous studies on this dataset (Mohammad and Pedersen, 2004), were not included. We apply a simple feature selection method: a feature X is selected if the conditional entropy $H(Y|X)$ is above a fixed threshold (1 bit) in the training set, and if X also occurs in the test set (note that no label information from the test data is used for this purpose).

5 Experiments

For both tasks we compare the performance of a supervised classifier, label propagation using the standard input features and either Euclidean or cosine distance, and LP using the output from a first-pass supervised classifier.

5.1 Lexicon acquisition task

5.1.1 First-pass classifier

For this task, the first-pass classifier is a multilayer perceptron (MLP) with the topology shown in Fig. 1. The input features are mapped to con-

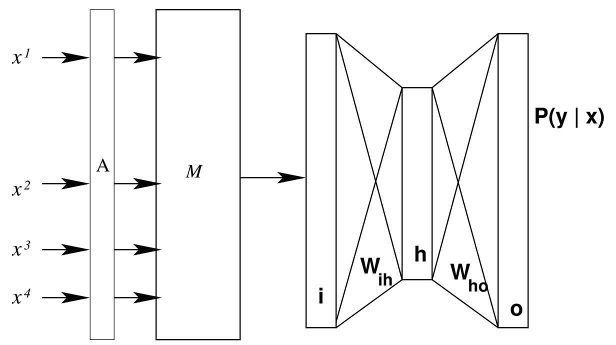


Figure 1: Architecture of first-pass supervised classifier (MLP) for lexicon acquisition

tinuous values by a discrete-to-continuous mapping layer M, which is itself learned during the MLP training process. This layer connects to the hidden layer **h**, which in turn is connected to the output layer **o**. The entire network is trained via backpropagation. The training criterion maximizes the regularized log-likelihood of the training data:

$$L = \frac{1}{n}\sum_{t=1}^{n} \log P(y_t|x_t, \theta) + R(\theta) \qquad (5)$$

The use of an additional continuous mapping layer is similar to the use of hidden continuous word representations in neural language modeling (Bengio et al., 2000) and yields better results than a standard 3-layer MLP topology.

Problems caused by data scarcity arise when some of the input features of the unlabeled words have never been seen in the training set, resulting in untrained, randomly-initialized values for those feature vector components. We address this problem by creating an approximation layer A that finds the known input feature vector x' that is most similar to x (by measuring the cosine similarity between the vectors). Then x_k is replaced with x'_k, resulting in vector $\hat{x} = \langle x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_f \rangle$ that has no unseen features and is closest to the original vector.

5.1.2 LP Setup

We use a dense graph approach. The WSJ set has a total of 44,492 words, therefore the P matrix that the algorithm requires would have $44,492 \times 44,492 \cong 2 \times 10^9$ elements. Due to the matrix size, we avoid the analytical solution of the LP problem, which requires inverting the P matrix, and choose the iterative approach described above (Sec. 2.1) instead. Convergence is stopped when the maximum relative difference between each cell of f and the corresponding cell of f' is less than 1%.

Also for data size reasons, we apply LP in chunks. While the training set stays in memory, the test data is loaded in fixed-size chunks, labeled, and discarded. This approach has yielded similar results for various chunk sizes, suggesting that chunking is a good approximation of whole-set label propagation.[1] LP in chunks is also amenable to parallelization: Our system labels different chunks in parallel.

We trained the α hyperparameter by three-fold cross-validation on the training data, using a geometric progression with limits 0.1 and 10 and ratio 2. We set fixed upper limits of edges between an unlabeled node and its labeled neighbors to 15, and between an unlabeled node and its unlabeled neighbors to 5. The approach of setting different limits among different kinds of nodes is also used in related work (Goldberg and Zhu, 2006).

For graph construction we tested: (a) the original discrete input representation with cosine distance; (b) the classifier output features (probability distributions) with the Jeffries-Matusita distance.

5.2 Combination optimization

The static parameters of the MLP (learning rate, regularization rate, and number of hidden units) were optimized for the LP step by 5-fold cross-validation on the training data. This process is important because overspecialization is detrimental to the combined system: an overspecialized first-pass classifier may output very confident but wrong predictions for unseen patterns, thus placing such samples at large distances from all correctly labeled samples. A strongly regularized neural network, by contrast, will output smoother probability distributions for unseen patterns. Such outputs also result in a smoother graph, which in turn helps the LP process. Thus, we found that a network with only 12 hidden units and relatively high $R(\theta)$ in Eq. 5 (10% of the weight value) performed best in combination with LP (at an insignificant cost in accuracy when used

[1] In fact, experiments have shown that performance tends to degrade for larger chunk sizes, suggesting that whole-set LP might be affected by "artifact" clusters that are not related to the labels.

as an isolated classifier).

5.2.1 Results

We first conducted an experiment to measure the smoothness of the underlying graph, $S(G)$, in the two LP experiments according to the following formula:

$$S(G) = \sum_{y_i \neq y_j, (i>n \vee j>n)} w_{ij} \qquad (6)$$

where y_i is the label of sample i. (Lower values are better as they reflect less affinity between nodes of different labels.) The value of $S(G)$ was in all cases significantly better on graphs constructed with our proposed technique than on graphs constructed in the standard way (see Table 1). Table 1 also shows the performance comparison between LP over the discrete representation and cosine distance ("LP"), the neural network itself ("NN"), and LP over the continuous representation ("NN+LP"), on all different subsets and for different training sizes. For scarce labeled data (5000 samples) the neural network, which uses a strictly supervised training procedure, is at a clear disadvantage. However, for a larger training set the network is able to perform more accurately than the LP learner that uses the discrete features directly. The third, combined technique outperforms the first two significantly.[2] The differences are more pronounced for smaller training set sizes. Interestingly, the LP is able to extract information from largely erroneous (noisy) distributions learned by the neural network.

5.3 Word Sense Disambiguation

We compare the performance of an SVM classifier, an LP learner using the same input features as the SVM, and an LP learner using the SVM outputs as input features. To analyze the influence of training set size on accuracy, we randomly sample subsets of the training data (25%, 50%, and 75%) and use the remaining training data plus the test data as unlabeled data, similarly to the procedure followed in related work (Niu et al., 2005). The results are averaged over five different random samplings. The samplings were chosen such that there was at least one sample for each label in the training set. SENSEVAL-3 sports multi-labeled samples and samples with the "unknown" label. We eliminate all samples labeled as unknown and retain only the first label for the multi-labeled instances.

[2]Significance was tested using a difference of proportions significance test; the significance level is 0.01 or smaller in all cases.

5.3.1 SVM setup

The use of SVM vs. MLP in this case was justified by the very small training data set. An MLP has many parameters and needs a considerable amount of data for effective training, so for this task with only on the order of 10^2 training samples per classifier, an SVM was deemed more appropriate. We use the SVM^{light} package to build a set of binary classifiers in a one-versus-all formulation of the multi-class classification problem. The features input to each SVM consist of the discrete features described above (Sec. 4.2) after feature selection. After training SVMs for each target label against the union of all others, we evaluate the SVM approach against the test set by using the winner-takes-all strategy: the predicted label corresponds to the SVM that outputs the largest value.

5.3.2 LP setup

Again we set up two LP systems: one using the original feature space (after feature selection, which benefited all of the tested systems) and one using the SVM outputs. Both use a cosine distance measure. The α parameter (see Eq. 1) is optimized through 3-fold cross-validation on the training set.

5.4 Combination optimization

Unlike MLPs, SVMs do not compute a smooth output distribution but base the classification decision on the sign of the output values. In order to smooth output values with a view towards graph construction we applied the following techniques:

1. *Combining SVM predictions and perfect feature vectors:* After training, the SVM actually outputs wrong label predictions for a small number ($\approx 5\%$) of training samples. These outputs could simply be replaced with the perfect SVM predictions (1 for the true class, -1 elsewhere) since the labels are known. However, the second-pass learner might actually benefit from the information contained in the misclassifications. We therefore linearly combine the SVM predictions with the "perfect" feature

Initial labels	Model	$S(G)$ avg.	Accuracy (%)					
			Set 1	Set 2	Set 3	Set 4	Set 5	Average
5000	NN	–	50.70	59.22	63.77	60.09	54.58	57.67 ± 4.55
	LP	451.54	58.37	59.91	60.88	62.01	59.47	60.13 ± 1.24
	NN+LP	409.79	58.03	63.91	66.62	65.93	57.76	**62.45 ± 3.83**
10000	NN	–	65.86	60.19	67.52	65.68	65.64	64.98 ± 2.49
	LP	381.16	58.27	60.04	60.85	61.99	62.06	60.64 ± 1.40
	NN+LP	315.53	69.36	64.73	69.50	70.26	67.71	**68.31 ± 1.97**
15000	NN	–	69.85	66.42	70.88	70.71	72.18	70.01 ± 1.94
	LP	299.10	58.51	61.00	60.94	63.53	60.98	60.99 ± 1.59
	NN+LP	235.83	70.59	69.45	69.99	71.20	73.45	**70.94 ± 1.39**

Table 1: Accuracy results of neural classification (NN), LP with discrete features (LP), and combined (NN+LP), over 5 random samplings of 5000, 10000, and 15000 labeled words in the WSJ lexicon acquisition task. $S(G)$ is the smoothness of the graph

vectors **v** that contain 1 at the correct label position and -1 elsewhere:

$$s'_i = \gamma s_i + (1 - \gamma)v_i \quad (7)$$

where s_i, s'_i are the i'th input and output feature vectors and γ a parameter fixed at 0.5.

2. *Biasing uninformative distributions:* For some training samples, although the predicted class label was correct, the outputs of the SVM were relatively close to one another, i.e. the decision was borderline. We decided to bias these SVM outputs in the right direction by using the same formula as in equation 7.
3. *Weighting by class priors:* For each training sample, a corresponding sample with the perfect output features was added, thus doubling the total number of labeled nodes in the graph. These synthesized nodes are akin to the "dongle" nodes (Goldberg and Zhu, 2006). The difference is that, while dongle nodes are only linked to one node, our artificial nodes are treated like any other node and as such can connect to several other nodes. The role of the artificial nodes is to serve as authorities during the LP process and to emphasize class priors.

5.4.1 Results

As before, we measured the smoothness of the graphs in the two label propagation setups and found that in all cases the smoothness of the graph produced with our method was better when compared to the graphs produced using the standard approach, as shown in Table 3, which also shows accuracy results for the SVM ("SVM" label), LP over the standard graph ("LP"), and label propagation over SVM outputs ("SVM+LP"). The latter system consistently performs best in all cases, although the most marked gains occur in the upper range of labeled samples percentage. The gain of the best data-driven LP over the knowledge-based LP is significant in the 100% and 75% cases.

#	System	Acc. (%)
1	htsa3 (Grozea, 2004)	72.9
2	IRST-kernels (Strapparava et al., 2004)	72.6
3	nusels (Lee et al., 2004)	72.4
4	SENSEVAL-3 contest baseline	55.2
5	Niu et al. (Niu et al., 2005) LP/J-S	70.3
6	Niu et al. LP/cosine	68.4
7	Niu et al. SVM	69.7

Table 2: Accuracy results of other published systems on SENSEVAL-3. 1-3 use syntactic features; 5-7 are directly comparably to our system.

For comparison purposes, Table 2 shows results of other published systems against the SENSEVAL corpus. The "htsa3", "IRST-kernels", and "nusels" systems were the winners of the SENSEVAL-3 contest and used extra input features (syntactic relations). The Niu et al. work (Niu et al., 2005) is most comparable to ours. We attribute the slightly higher performance of our SVM due to our feature selection process. The LP/cosine system is a system similar to our LP system using the discrete features, and the LP/Jensen-Shannon system is also similar but uses a distance measure derived from Jensen-Shannon divergence.

6 Conclusions

We have presented a data-driven graph construction technique for label propagation that utilizes a first-

Initial labels	Model	$S(G)$ avg.	Accuracy (%)					
			Set 1	Set 2	Set 3	Set 4	Set 5	Average
25%	SVM	–	62.94	62.53	62.69	63.52	62.99	62.93 ± 0.34
	LP	44.71	63.27	61.84	63.26	62.96	63.30	62.93 ± 0.56
	SVM+LP	39.67	63.39	63.20	63.95	63.68	63.91	**63.63 ± 0.29**
50%	SVM	–	67.90	66.75	67.57	67.44	66.79	67.29 ± 0.45
	LP	33.17	67.84	66.57	67.35	66.52	66.35	66.93 ± 0.57
	SVM+LP	24.19	67.95	67.54	67.93	68.21	68.11	**67.95 ± 0.23**
75%	SVM	–	69.54	70.19	68.75	69.80	68.73	69.40 ± 0.58
	LP	29.93	68.87	68.65	68.58	68.42	67.19	68.34 ± 0.59
	SVM+LP	16.19	69.98	70.05	69.69	70.38	68.94	**69.81 ± 0.49**
100%	SVM	–						70.74
	LP	21.72						69.69
	SVM+LP	13.17						**71.72**

Table 3: Accuracy results of support vector machine (SVM), label propagation over discrete features (LP), and label propagation over SVM outputs (SVM+LP), each trained with 25%, 50%, 75% (5 random samplings each), and 100% of the train set. The improvements of SVM+LP are significant over LP in the 75% and 100% cases. $S(G)$ is the graph smoothness

pass supervised classifier. The outputs from this classifier (especially when optimized for the second-pass learner) were shown to serve as a better representation for graph-based semi-supervised learning. Classification results on two learning tasks showed significantly better performance compared to LP using standard graph construction and the supervised classifier alone.

Acknowledgments This work was funded by NSF under grant no. IIS-0326276. Any opinions, findings and conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of this agency.

References

E. Alpaydin and C. Kaynak. 1998. Cascading classifiers. *Kybernetika*, 34:369–374.

Balcan et al. 2005. Person identification in webcam images. In *ICML Workshop on Learning with Partially Classified Training Data*.

Y. Bengio, R. Ducharme, and P. Vincent. 2000. A neural probabilistic language model. In *NIPS*.

J. Chen, D. Ji, C.L. Tan, and Z. Niu. 2006. Relation Extraction Using Label Propagation Based Semi-supervised Learning. In *Proceedings of ACL*, pages 129–136.

A. Goldberg and J. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL Workshop on Graph-based Algorithms for Natural Language Processing*.

C. Grozea. 2004. Finding optimal parameter settings for high performance word sense disambiguation. *Proceedings of Senseval-3 Workshop*.

A. Haghighi, A. Ng, and C.D. Manning. 2005. Robust textual inference via graph matching. *Proceedings of EMNLP*.

Y.K. Lee and H.T. Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP*, pages 41–48.

Y.K. Lee, H.T. Ng, and T.K. Chia. 2004. Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. *SENSEVAL-3*.

R. Mihalcea, T. Chklovski, and A. Killgariff. 2004. The Senseval-3 English Lexical Sample Task. In *Proceedings of ACL/SIGLEX Senseval-3*.

R. Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of HLT/EMNLP*, pages 411–418.

S. Mohammad and T. Pedersen. 2004. Complementarity of Lexical and Simple Syntactic Features: The SyntaLex Approach to Senseval-3. *Proceedings of the SENSEVAL-3*.

Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning. In *ACL '05*.

J. Otterbacher, G. Erkan, and D.R. Radev. 2005. Using Random Walks for Question-focused Sentence Retrieval. *Proceedings of HLT/EMNLP*, pages 915–922.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, pages 271–278.

T. Qin, T.-Y. Liu, X.-D. Zhang, W.-Y. Ma, and H.-J. Zhang. 2005. Subspace clustering and label propagation for active feedback in image retrieval. In *MMM*, pages 172–179.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

C. Strapparava, A. Gliozzo, and C. Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation: IRST at SENSEVAL-3. *Proc. of SENSEVAL-3*, pages 229–234.

J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W. Noble. 2003. Semi-supervised protein classification using cluster kernels.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02.

Xiaojin Zhu. 2005. *Semi-Supervised Learning with Graphs*. Ph.D. thesis, Carnegie Mellon University. CMU-LTI-05-192.

Is Question Answering Better Than Information Retrieval? Towards a Task-Based Evaluation Framework for Question Series

Jimmy Lin
College of Information Studies
Department of Computer Science
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742, USA
jimmylin@umd.edu

Abstract

This paper introduces a novel evaluation framework for question series and employs it to explore the effectiveness of QA and IR systems at addressing users' information needs. The framework is based on the notion of recall curves, which characterize the amount of relevant information contained within a fixed-length text segment. Although it is widely assumed that QA technology provides more efficient access to information than IR systems, our experiments show that a simple IR baseline is quite competitive. These results help us better understand the role of NLP technology in QA systems and suggest directions for future research.

1 Introduction

The emergence of question answering (QA) has been driven to a large extent by its intuitive appeal. Instead of "hits", QA technology promises to deliver "answers", obviating the user from the tedious task of sorting through lists of potentially-relevant documents. The success of factoid QA systems, particularly in the NIST-sponsored TREC evaluations (Voorhees, 2003), has reinforced the perception about the superiority of QA systems over traditional IR engines.

However, is QA really better than IR? This work challenges existing assumptions and critically examines this question, starting with the development of a novel evaluation framework that better models user tasks and preferences. The framework is then applied to compare top TREC QA systems against an off-the-shelf IR engine. Surprisingly, experiments show that the IR baseline is quite competitive. These results help us better understand the added value of NLP technology in QA systems, and are also useful in guiding future research.

2 Evolution of QA Evaluation

Although most question answering systems rely on information retrieval technology, there has always been the understanding that NLP provides significant added value beyond simple IR. Even the earliest open-domain factoid QA systems, which can be traced back to the late nineties (Voorhees and Tice, 1999), demonstrated the importance and impact of linguistic processing. Today's top systems deploy a wide range of advanced NLP technology and can answer over three quarters of factoid questions in an open domain (Voorhees, 2003). However, present QA evaluation methodology does not take into account two developments, discussed below.

First, despite trends to the contrary in TREC evaluations, users don't actually like or want exact answers. Most question answering systems are designed to pinpoint the exact named entity (person, date, organization, etc.) that answers a particular question—and the development of such technology has been encouraged by the setup of the TREC QA tracks. However, a study by Lin et al. (2003) shows that users actually prefer answers embedded within some sort of context, e.g., the sentence or the paragraph that the answer was found in. Context pro-

3. Hale Bopp comet	
1. fact	When was the comet discovered?
2. fact	How often does it approach the earth?
3. list	In what countries was the comet visible on its last return?
4. other	

68. Port Arthur Massacre	
1. fact	Where is Port Arthur?
2. fact	When did the massacre occur?
3. fact	What was the final death toll of the massacre?
4. fact	Who was the killer?
5. fact	What was the killer's nationality?
6. list	What were the names of the victims?
7. list	What were the nationalities of the victims?
8. other	

Table 1: Sample question series.

vides a means by which the user can establish the credibility of system responses and also provides a vehicle for "serendipitous knowledge discovery"—finding answers to related questions. As the early TRECs have found (Voorhees and Tice, 1999), locating a passage that contains an answer is considerably easier than pinpointing the exact answer. Thus, real-world user preferences may erode the advantage that QA has over IR techniques such as passage retrieval, e.g., (Zobel et al., 1995; Tellex et al., 2003).

Second, the focus of question answering research has shifted away from isolated factoid questions to more complex information needs embedded within a broader context (e.g., a user scenario). Since 2004, the main task at the TREC QA tracks has consisted of question series organized around topics (called "targets")—which can be people, organizations, entities, or events (Voorhees, 2004; Voorhees, 2005). Questions in a series inquire about different facets of a target, but are themselves either factoid or list questions. In addition, each series contains an explicit "other" question (always the last one), which can be paraphrased as "Tell me other interesting things about this target that I don't know enough to ask directly." See Table 1 for examples of question series. Separately, NIST has been exploring other types of complex information needs, for example, the relationship task in TREC 2005 and the ciQA (complex, interactive Question Answering) task in TREC 2006 (Dang et al., 2006). One shared feature of these complex questions is that they cannot be answered by simple named entities. Answers usually span passages, which makes the task very similar to the query-focused summarization task in DUC (Dang, 2005). On these tasks, it is unclear whether QA systems actually outperform baseline IR methods. As one bit of evidence, in TREC 2003, a simple IR-based sentence ranker outperformed all but the best system on definition questions, the precursor to current "other" questions (Voorhees, 2003).

We believe that QA evaluation methodology has lagged behind these developments and does not adequately characterize the performance of current systems. In the next section, we present an evaluation framework that takes into account users' desire for context and the structure of more complex QA tasks. Focusing on question series, we compare the performance of top TREC systems to a baseline IR engine using this evaluation framework.

3 An Evaluation Framework

Question series in TREC represent an attempt at modeling information-seeking dialogues between a user and a system (Kato et al., 2004). Primarily because dialogue systems are difficult to evaluate, NIST has adopted a setup in which individual questions are evaluated in isolation—this implicitly models a user who types in a question, receives an answer, and then moves on to the next question in the series. Component scores are aggregated using a weighted average, and no attempt is made to capture dependencies across different question types.

Simultaneously acknowledging the challenges in evaluating dialogue systems and recognizing the similarities between complex QA and query-focused summarization, we propose an alternative framework for QA evaluation that considers the quality of system responses as a whole. Instead of generating individual answers to each question, a system might alternatively produce a segment of text (i.e., a summary) that attempts to answer *all* the questions. This slightly different conception of QA brings it into better alignment with recent trends in multi-document summarization, which may yield previously untapped synergies (see Section 7).

To assess the quality of system responses, we adopt the nugget-based methodology used previously for many types of complex questions (Voorhees, 2003), which shares similarities with the pyramid evaluation scheme used in summarization (Nenkova and Passonneau, 2004). A nugget can be described as an "atomic fact" that addresses an aspect of an information need. Instead of the standard nugget F-score, which hides important tradeoffs between precision and recall, we propose to measure nugget recall as a function of response length. The goal is to quantify the number of relevant facts that a user will have encountered after reading a particular amount of text. Intuitively, we wish to model how quickly a hypothetical user could "learn" about a topic by reading system responses.

Within this framework, we compared existing TREC QA systems against an IR baseline. Processed outputs from the top-ranked, second-ranked, third-ranked, and median runs in TREC 2004 and TREC 2005 were compared to a baseline IR run generated by Lucene, an off-the-shelf open-source IR engine. Our experiments focused on factoid and "other" questions; as the details differ for these two types, we describe each separately and then return to a unified picture.

4 Factoid Series

Our first set of experiments focuses on the factoid questions within a series. In what follows, we describe the data preparation process, the evaluation methodology, and experimental results.

4.1 Data Preparation

We began by preparing answer responses from the top-ranked, second-ranked, third-ranked, and median runs from TREC 2004 and TREC 2005.[1] Consider the third-ranked run from TREC 2004 as a running example; for the two factoid questions in target 3 (Table 1), the system answers were "July 22, 1995" and "4,200 years" (both correct).

Since Lin et al. (2003) suggest that users prefer answers situated within some sort of context, we projected these exact answers onto their source sentences. This was accomplished by selecting the first sentence in the source document (drawn from the AQUAINT corpus) that contains the answer string.[2] In our example, this procedure yielded the following text segment:

> The comet was named after its two observers—two amateur astronomers in the United States who discovered it on July 22, 1995. Its visit to the solar system—just once every 4,200 years, will give millions of people a rare heavenly treat when it reaches its full brightness next year.

Since projected sentences are simply concatenated, the responses often exhibit readability problems (although by chance this particular response is relatively coherent). Nevertheless, one might imagine that such output forms the basis for generating coherent query-focused summaries with sentence-rewrite techniques, e.g., (Barzilay et al., 1999). In this work, we set aside problems with fluency since our evaluation framework is unable to measure this (desirable) characteristic.

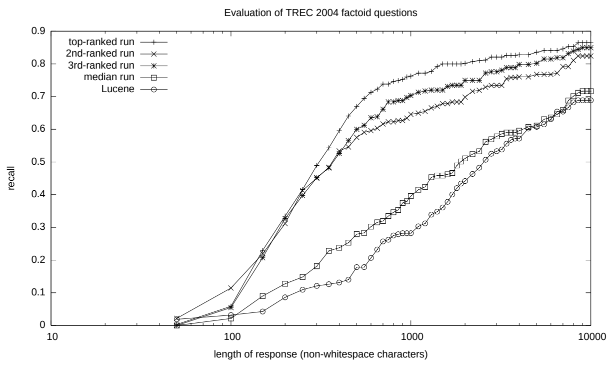
System responses were prepared for four runs from TREC 2004 and four runs from TREC 2005 in the manner described above. As a baseline, we employed Lucene to retrieve the top 100 documents from the AQUAINT corpus using the target as the query (in our example, "Hale Bopp comet"). From the result set, we retained all sentences that contain at least a term from the target. Sentence order within each document and across the ranked list was preserved. Answer responses for this baseline condition were limited to 10,000 characters. Following TREC convention, all character counts include only non-whitespace characters. Finally, since responses prepared from TREC runs were significantly shorter than this baseline condition, the baseline Lucene response was appended to the end of each TREC run to fill a quota of 10,000 characters.

4.2 Evaluation Methodology

Our evaluation framework is designed to measure the amount of useful information contained in a system response. For factoid series, this can be quan-

[1] In cases where teams submitted multiple runs, we considered only the best performing of each.

[2] As a backoff, if the exact answer string is not found in the text, the sentence with the most terms in common with the answer string is selected.



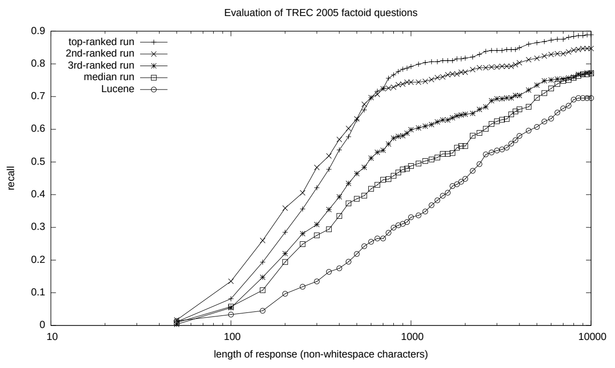


Figure 1: Factoid recall curves for runs from TREC 2004 (left) and TREC 2005 (right).

Run	**2004**	**2005**
top-ranked run	0.770	0.713
2nd-ranked run	0.643	0.666
3rd-ranked run	0.626	0.326
median run	0.170	0.177

Table 2: Official scores of selected TREC 2004 and TREC 2005 factoid runs.

tified by recall—the fraction of questions within a series whose answers could be found within a given passage. By varying the passage length, we can characterize systems in terms of recall curves that represent how quickly a hypothetical user can "learn" about the target. Below, we describe the implementation of such a metric.

First, we need a method to automatically determine if an answer string is contained within a segment of text. For this, regular expression answer patterns distributed by NIST were employed—they have become a widely-accepted evaluation tool.

Second, we must determine when a fact is "acquired" by our hypothetical user. Since previous studies suggest that context is needed to interpret an answer, we assess system output on a sentence-by-sentence basis. In our example, the lengths of the two sentences are 105 and 130 characters, respectively. Thus, for this series, we obtain a recall of 0.5 at 105 characters and 1.0 at 235 characters.

Finally, we must devise a method for aggregating across different question series to factor out variations. We accomplish this through interpolation, much in the same way that precision–recall curves are plotted in IR experiments. First, all lengths are interpolated to their nearest larger fifty character increment. In our case, they are 150 and 250. Once this is accomplished for each question series, we can directly average across all question series at each length increment. Plotting these points gives us a recall-by-length performance curve.

4.3 Results

Results of our evaluation are shown in Figure 1, for TREC 2004 (left) and TREC 2005 (right). These plots have a simple interpretation—curves that rise faster and higher represent "better" systems. The "knee" in some of the curves indicate approximately the length of the original system output (recall that the baseline Lucene run was appended to each TREC run to produce responses of equal lengths). For reference, official factoid scores of the same runs are shown in Table 2.

Results from TREC 2004 are striking: while the top three systems appear to outperform the baseline IR run, it is unclear if the median system is better than Lucene, especially at longer response lengths. This suggests that if a user wanted to obtain answers to a series of factoid questions about a topic, using the median QA system isn't any more efficient than simply retrieving a few articles using an IR engine and reading them. Turning to the 2005 results, the median system fares better when compared to the IR baseline, although the separation between the top and median systems has narrowed.

In the next two sections, we present additional experiments on question series. A detailed analysis is saved for Section 7.

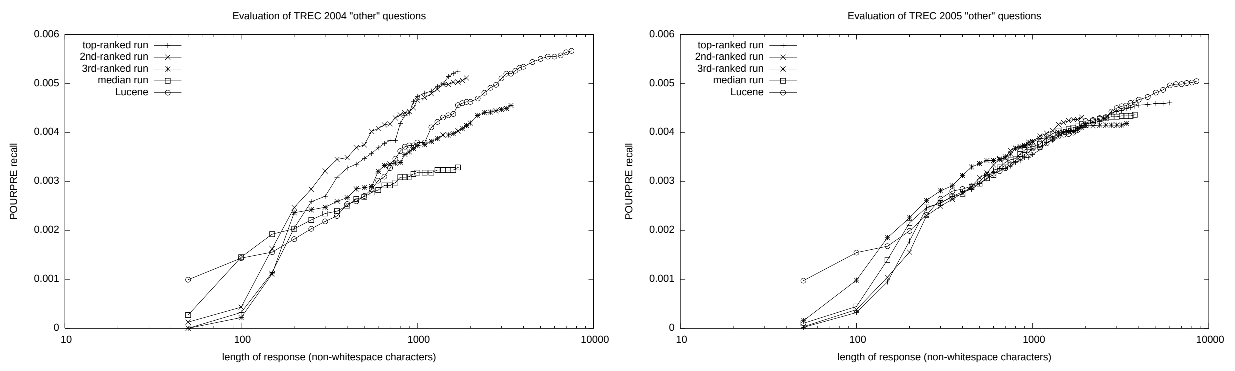


Figure 2: POURPRE recall curves for "other" runs from TREC 2004 (left) and TREC 2005 (right).

Run	2004	2005
top-ranked run	0.460	0.248
2nd-ranked run	0.404	0.232
3rd-ranked run	0.367	0.228
median run	0.197	0.152

Table 3: Official scores of selected TREC 2004 and TREC 2005 "other" runs.

5 "Other" Questions

Our second set of experiments examine the performance of TREC systems on "other" questions. Once again, we selected the top-ranked, second-ranked, third-ranked, and median runs from TREC 2004 and TREC 2005. Since system submissions were already passages, no additional processing was necessary. The IR baseline was exactly the same as the run used in the previous experiment. Below, we describe the evaluation methodology and results.

5.1 Evaluation Methodology

The evaluation of "other" questions closely mirrors the procedure developed for factoid series. We employed POURPRE (Lin and Demner-Fushman, 2005), a recently developed method for automatically evaluating answers to complex questions. The metric relies on *n*-gram overlap as a surrogate for manual nugget matching, and has been shown to correlate well with official human judgments. We modified the POURPRE scoring script to return only the nugget recall (of vital nuggets only).

Formally, systems' responses to "other" questions consist of unordered sets of answer strings. We decided to break each system's response into individual answer strings and compute nugget recall on a string-by-string basis. Since these answer strings are for the most part sentences, results are comparable to the factoid series experiments. Taking answer strings as the basic response unit also makes sense because it respects segment boundaries that are presumably meaningful to the original systems.

Computing POURPRE recall at different response lengths yielded an uninterpolated data series for each topic. Results across topics were aggregated in the same manner as the factoid series: first by interpolating to the nearest larger fifty-character increment, and then averaging all topics across each length increment.[3]

5.2 Results

Results of our experiment are shown in Figure 2. For reference, the official nugget F-scores of the TREC runs are shown in Table 3. Most striking is the observation that the baseline Lucene run is highly competitive with submitted TREC systems. For TREC 2004, it appears that the IR baseline outperforms all but the top two systems at higher recall levels. For TREC 2005, differences between all the analyzed runs are difficult to distinguish. Although scores of submitted runs in TREC 2005 were more tightly clustered, the strong baseline IR performance is surprising. For "other" questions, it doesn't appear that QA is better than IR!

We believe that relative differences in QA and IR

[3]It is worth noting that this protocol treats the answer strings as if they were ordered—but we do not believe this has an impact on the results or our conclusions.

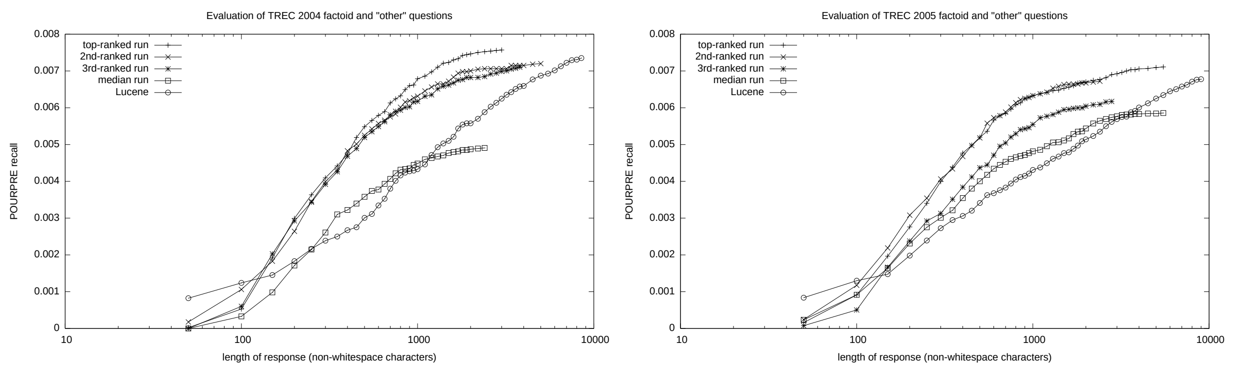


Figure 3: POURPRE recall curves for runs from TREC 2004 (left) and TREC 2005 (right), combining both factoid and "other" questions.

performance between the 2004 and 2005 test sets can be attributed to the nature of the targets. In TREC 2005, allowable semantic categories of targets were expanded to include events such as "Miss Universe 2000 crowned", which by their very nature are narrower in scope. This, combined with many highly-specific targets, meant that the corpus contained fewer topically-relevant documents for each target to begin with. As a result, an IR-based sentence extraction approach performs quite well—this explanation is consistent with the observations of Lin and Demner-Fushman (2006).

6 Combining Question Types

In the previous two sections, factoid and "other" questions were examined in isolation, which ignores their complementary role in supplying information about a target. To provide a more complete picture of system performance, we devised a method by which both question types can be evaluated together.

At the conceptual level, there is little difference between factoid and "other" questions. The first type asks for explicit facts, while the second type asks for facts that the user didn't know enough to ask about directly. We can unify the evaluation of both types by treating regular expression factoid patterns as if they were (vital) nuggets. Many patterns don't contain any special symbols, and read quite like nugget descriptions already. For others, we manually converted regular expressions into plain text, e.g., "(auto|car) crash" becomes "auto car crash".

To validate this method, we first evaluated factoid series using POURPRE, with nugget descriptions prepared from answer patterns in the manner described above. For both TREC 2004 and TREC 2005, we did not notice any qualitative differences in the results, suggesting that factoid answers can indeed be treated like nuggets.

We then proceeded to evaluate both factoid and "other" questions together using the above procedure. Runs were prepared by appending the 1st "other" run to the 1st factoid run, the 2nd "other" run to the 2nd factoid run, etc.[4] The Lucene baseline run remained the same as before.

Plots of POURPRE recall by answer length are shown in Table 3. These graphs provide a more complete picture of QA performance on question series. The same trends observed in the two previous experiments are seen here also: it does not appear that the median run in TREC 2004 performs any better than the IR baseline. Considering the TREC 2005 runs, the IR baseline remains surprisingly competitive.

Note that integration of list questions, the third component of question series, remains a challenge. Whereas the answer to a factoid question can be naturally viewed as a vital nugget describing the target, the relative importance of a single answer instance to a list question cannot be easily quantified. We leave this issue for future work.

7 Discussion

It can be argued that quantitative evaluation is the single most important driver for advancing the state

[4]Note that we're mixing sections from different runs, so these do not correspond to any actual TREC submissions.

of the art in language processing technology today. As a result, evaluation metrics and methodologies need to be carefully considered to insure that they provide proper guidance to researchers. Along these lines, this paper makes two arguments: that recall curves better capture aspects of complex QA tasks than the existing TREC evaluation metrics; and that this novel evaluation framework allows us to explore the relationship between QA and IR technology in a manner not possible before.

7.1 Advantages of Recall Curves

We see several advantages to the evaluation framework introduced here, beyond those already discussed in Sections 2 and 3.

Previously, QA and IR techniques were not directly comparable since they returned different response units. To make evaluation even more complex, different types of questions (e.g., factoid vs. "other") require different metrics—in TREC, these incomparable values were then aggregated based on arbitrary weights to produce a final composite score. By noting similarities between factoid answers and nuggets, we were able to develop a unified evaluation framework for factoid and "other" questions. By emphasizing the similarities between complex QA and summarization, it becomes possible to compare QA and IR technology directly—this work provides a point of reference much in the same way that IR-based sentence extraction has served as a starting point for summarization research, e.g., (Goldstein et al., 1999).

In addition, characterizing system performance in terms of recall curves allows researchers to compare the effectiveness of systems under different task models. Measuring recall at short response lengths might reflect time-constrained scenarios, e.g., producing an action-oriented report with a 30-minute deadline. Measuring recall at longer response lengths might correspond to in-depth research, e.g., writing a summary article due by the end of the day. Recall curves are able to capture potential system tradeoffs that might otherwise be hidden in single-point metrics.

7.2 Understanding QA and IR

Beyond answering a straightforward question, the results of our experiments yield insights about the relationship between QA and IR technology.

Most question answering systems today employ a two-stage architecture: IR techniques are first used to select a candidate set of documents (or alternatively, passages, sentences, etc.), which is then analyzed by more sophisticated NLP techniques. For factoids, analysis usually involves named-entity recognition using some sort of answer type ontology; for "other" questions, analysis typically includes filtering for definitions based on surface patterns and other features. The evaluation framework described in this paper is able to isolate the performance contribution of this second NLP stage—which corresponds to the difference between the baseline IR and QA recall curves.

For factoid questions, NLP technology provides a lot of added value: the set of techniques developed for pinpointing exact answers allows users to acquire information more quickly than they otherwise could with an IR system (shown by Figure 1). The added value of NLP techniques for answering "other" questions is less clear—in many instances, those techniques do not appear to be contributing much (shown by Figure 2). Whereas factoid QA technology is relatively mature, researchers have made less progress in developing general techniques for answering complex questions.

Our experiments also illuminate when exactly QA works. For short responses, there is little difference between QA and IR, or between all QA systems for that matter, since it is difficult to cram much information into a short response with current (extractive) technology. For extremely long responses, the advantages provided by the best QA systems are relatively small, since there's an upper limit to their accuracy (and researchers have yet to develop a good backoff strategy). In the middle range of response lengths is where QA technology really shines—where a user can much more effectively gather knowledge using a QA system.

7.3 Implications for Future Research

Based on the results presented here, we suggest two future directions for the field of question answering.

First, we believe there is a need to focus on answer generation. High-precision answer extraction alone isn't sufficient to address users' complex information needs—information nuggets must be synthesized and presented for efficient human consumption. The coherence and fluency of system responses should be factored into the evaluation methodology as well. In this regard, QA researchers have much to learn from the summarization community, which has already grappled with these issues.

Second, more effort is required to developed task-based QA evaluations. The "goodness" of answers can only be quantified with respect to a task—examples range from winning a game show (Clarke et al., 2001) to intelligence gathering (Small et al., 2004). It is impossible to assess the real-world impact of QA technology without considering how such systems will be used to solve human problems. Our work takes a small step in this direction.

8 Conclusion

Is QA better than IR? The short answer, somewhat to our relief, is *yes*. But this work provides more than a simple affirmation. We believe that our contributions are two-fold: a novel framework for evaluating QA systems that more realistically models user tasks and preferences, and an exploration of QA and IR performance within this framework that yields new insights about these two technologies. We hope that these results are useful in guiding the development of future question answering systems.

9 Acknowledgments

This work has been supported in part by DARPA contract HR0011-06-2-0001 (GALE), and has benefited from discussions with Ellen Voorhees and Hoa Dang. I would also like to thank Esther and Kiri for their loving support.

References

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proc. of ACL 1999*.

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001. Exploiting redundancy in question answering. In *Proc. of SIGIR 2001*.

Hoa Trang Dang, Jimmy Lin, and Diane Kelly. 2006. Overview of the TREC 2006 question answering track. In *Proc. of TREC 2006*.

Hoa Dang. 2005. Overview of DUC 2005. In *Proc. of DUC 2005*.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Proc. of SIGIR 1999*.

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 *Proc. of the HLT-NAACL 2004 Workshop on Pragmatics of Question Answering*.

Jimmy Lin and Dina Demner-Fushman. 2005. Automatically evaluating answers to definition questions. In *Proc. of HLT/EMNLP 2005*.

Jimmy Lin and Dina Demner-Fushman. 2006. Will pyramids built of nuggets topple over? In *Proc. of HLT/NAACL 2006*.

Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. 2003. What makes a good answer? The role of context in question answering. In *Proc. of INTERACT 2003*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proc. of HLT/NAACL 2004*.

Sharon Small, Tomek Strzalkowski, Ting Liu, Sean Ryan, Robert Salkin, Nobuyuki Shimizu, Paul Kantor, Diane Kelly, Robert Rittman, and Nina Wacholder. 2004. HITIQA: towards analytical question answering. In *Proc. of COLING 2004*.

Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proc. of SIGIR 2003*.

Ellen M. Voorhees and Dawn M. Tice. 1999. The TREC-8 question answering track evaluation. In *Proc. of TREC-8*.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Proc. of TREC 2003*.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 question answering track. In *Proc. of TREC 2004*.

Ellen M. Voorhees. 2005. Using question series to evaluate question answering system effectiveness. In *Proc. of HLT/EMNLP 2005*.

Justin Zobel, Alistair Moffat, and Ross Wilkinson Ron Sacks-Davis. 1995. Efficient retrieval of partial documents. *IPM*, 31(3):361–377.

A Case for Shorter Queries, and Helping Users Create Them

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

Information retrieval systems are frequently required to handle long queries. Simply using all terms in the query or relying on the underlying retrieval model to appropriately weight terms often leads to ineffective retrieval. We show that rewriting the query to a version that comprises a small subset of appropriate terms from the original query greatly improves effectiveness. Targeting a demonstrated potential improvement of almost 50% on some *difficult* TREC queries and their associated collections, we develop a suite of automatic techniques to re-write queries and study their characteristics. We show that the shortcomings of automatic methods can be ameliorated by some *simple* user interaction, and report results that are on average 25% better than the baseline.

1 Introduction

Query expansion has long been a focus of information retrieval research. Given an arbitrary short query, the goal was to find and include additional related and suitably-weighted terms to the original query to produce a more effective version. In this paper we focus on a complementary problem – *query re-writing*. Given a long query we explore whether there is utility in modifying it to a more concise version such that the original information need is still expressed.

The Y!Q beta[1] search engine allows users to select large portions of text from documents and issue them as queries. The search engine is designed to encourage users to submit long queries such as this example from the web site *"I need to know the gas mileage for my Audi A8 2004 model"*. The motivation for encouraging this type of querying is that longer queries would provide more information in the form of context (Kraft et al., 2006), and this additional information could be leveraged to provide a better search experience. However, handling such long queries is a challenge. The use of all the terms from the user's input can rapidly narrow down the set of matching documents, especially if a boolean retrieval model is adopted. While one would expect the underlying retrieval model to appropriately assign weights to different terms in the query and return only relevant content, it is widely acknowledged that models fail due to a variety of reasons (Harman and Buckley, 2004), and are not suited to tackle every possible query.

Recently, there has been great interest in personalized search (Teevan et al., 2005), where the query is modified based on a user's profile. The profile usually consists of documents previously viewed, web sites recently visited, e-mail correspondence and so on. Common procedures for using this large amount of information usually involve creating huge query vectors with some sort of term-weighting mechanism to favor different portions of the profile.

The queries used in the TREC ad-hoc tracks consist of title, description and narrative sections, of progressively increasing length. The title, of length

[1] http://yq.search.yahoo.com/

ranging from a single term to four terms is considered a concise query, while the description is considered a longer version of the title expressing the same information need. Almost all research on the TREC ad-hoc retrieval track reports results using only the title portion as the query, and a combination of the title and description as a separate query. Most reported results show that the latter is more effective than the former, though in the case of some hard collections the opposite is true. However, as we shall show later, there is tremendous scope for improvement. Formulating a shorter query from the description can lead to significant improvements in performance.

In the light of the above, we believe there is great utility in creating query-rewriting mechanisms for handling long queries. This paper is organized in the following way. We start with some examples and explore ways by which we can create concise high-quality reformulations of long queries in Section 2. We describe our baseline system in Section 3 and motivate our investigations with experiments in Section 4. Since automatic methods have shortfalls, we present a procedure in Section 5 to involve users in selecting a good shorter query from a small selection of alternatives. We report and discuss the results of this approach in Section 6. Related work is presented in Section 7. We wrap up with conclusions and future directions in Section 8.

2 Selecting sub-queries

Consider the following query:

Define Argentine and British international relations.

When this query was issued to a search engine, the average precision (AP, Section 3) of the results was 0.424. When we selected subsets of terms (sub-queries) from the query, and ran them as distinct queries, the performance was as shown in Table 1. It can be observed that there are seven different ways of re-writing the original query to attain better performance. The best query, also among the shortest, did not have a natural-language flavor to it. It however had an effectiveness almost 50% more than the original query. This immense potential for improvement by query re-writing is the motivation for this paper.

Query	AP
....	....
international relate	0.000
define international relate	0.000
....	....
define argentina	0.123
international relate argentina	0.130
define relate argentina	0.141
relate argentina	0.173
define britain international relate argentina	*0.424*
define britain international argentina	0.469
britain international relate argentina	0.490
define britain relate argentina	0.494
britain international argentina	0.528
define britain argentina	0.546
britain relate argentina	0.563
britain argentina	0.626

Table 1: The results of using all possible subsets (excluding singletons) of the original query as queries. The query terms were stemmed and stopped.

Analysis of the terms in the sub-queries and the relationship of the sub-queries with the original query revealed a few interesting insights that had potential to be leveraged to aid sub-query selection.

1. Terms in the original query that a human would consider vital in conveying the type of information desired were missing from the best sub-queries. For example, the best sub-query for the example was *britain argentina*, omitting any reference to international relations. This also reveals a mismatch between the user's query and the way terms occurred in the corpus, and suggests that an approximate query could at times be a better starting point for search.

2. The sub-query would often contain *only* terms that a human would consider vital to the query while the original query would also (naturally) contain them, albeit weighted lower with respect to other terms. This is a common problem (Harman and Buckley, 2004), and the focus of efforts to isolate the key *concept* terms in queries (Buckley et al., 2000; Allan et al., 1996).

3. Good sub-queries were missing many of the noise terms found in the original query. Ideally the retrieval model would weight them lower, but dropping them completely from the query appeared to be more effective.

4. Sub-queries a human would consider as an incomplete expression of information need sometimes performed better than the original query. Our example illustrates this point.

Given the above empirical observations, we explored a variety of procedures to refine a long query into a shorter one that retained the key terms. We expected the set of terms of a good sub-query to have the following properties.

A. *Minimal Cardinality*: Any set that contains more than the minimum number of terms to retrieve relevant documents could suffer from concept drift.

B. *Coherency*: The terms that constitute the sub-query should be coherent, i.e. they should buttress each other in representing the information need. If need be, terms that the user considered important but led to retrieval of non-relevant documents should be dropped.

Some of the sub-query selection methods we explored with these properties in mind are reported below.

2.1 Mutual Information

Let X and Y be two random variables, with joint distribution $P(x, y)$ and marginal distributions $P(x)$ and $P(y)$ respectively. The mutual information is then defined as:

$$I(X;Y) = \sum_x \sum_y p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{1}$$

Intuitively, mutual information measures the information about X that is shared by Y. If X and Y are independent, then X contains no information about Y and vice versa and hence their mutual information is zero. Mutual Information is attractive because it is not only easy to compute, but also takes into consideration corpus statistics and semantics. The mutual information between two terms (Church and Hanks, 1989) can be calculated using Equation 2.

$$I(x,y) = log \frac{\frac{n(x,y)}{N}}{\frac{n(x)}{N}\frac{n(y)}{N}} \tag{2}$$

$n(x, y)$ is the number of times terms x and y occurred within a term window of 100 terms across the corpus, while $n(x)$ and $n(y)$ are the frequencies of x and y in the collection of size N terms.

To tackle the situation where we have an arbitrary number of variables (terms) we extend the two-variable case to the multivariate case. The extension, called multivariate mutual information (MVMI) can be generalized from Equation 1 to:

$$I(X_1; X_2; X_3; ...; X_N) = \sum_{i=1}^{N} (-1)^{i-1} \sum_{X \subset (X_1, X_2, X_3, ..., X_N), |X|=k} H(X) \tag{3}$$

The calculation of multivariate information using Equation 3 was very cumbersome, and we instead worked with the approximation (Kern et al., 2003) given below.

$$I(X_1; X_2; X_3; ...; X_N) = \tag{4}$$

$$\sum_{i,j=\{1,2,3,...,N; i \neq j\}} I(X_i; X_j) \tag{5}$$

For the case involving multiple terms, we calculated MVMI as the sum of the pair-wise mutual information for all terms in the candidate sub-query. This can be also viewed as the creation of a completely connected graph $G = (V, E)$, where the vertices V are the terms and the edges E are weighted using the mutual information between the vertices they connect.

To select a score representative of the quality of a sub-query we considered several options including the sum, average, median and minimum of the edge weights. We performed experiments on a set of candidate queries to determine how well each of these measures tracked AP, and found that the average worked best. We refer to the sub-query selection procedure using the average score as **Average**.

2.2 Maximum Spanning Tree

It is well-known that an average is easily skewed by outliers. In other words, the existence of one or more terms that have low mutual information with every other term could potentially distort results. This problem could be further compounded by the fact that mutual information measured using Equation 2 could have a negative value. We attempted to tackle this problem by considering another measure that involved creating a maximum spanning tree (MaxST) over the fully connected graph G, and using the weight of the identified tree as a measure representative of the candidate query's quality (Rijsbergen, 1979). We used Kruskal's minimum spanning tree (Cormen et al., 2001) algorithm after negating the edge weights to obtain a MaxST. We refer to the sub-query selection procedure using the weight of the maximum spanning tree as **MaxST**.

2.3 Named Entities

Named entities (names of persons, places, organizations, dates, etc.) are known to play an important anchor role in many information retrieval applications. In our example from Section 2, sub-queries without *Britain* or *Argentina* will not be effective even though the mutual information score of the other two terms *international* and *relations* might indicate otherwise. We experimented with another version of sub-query selection that considered only sub-queries that retained at least one of the named entities from the original query. We refer to the variants that retained named entities as **NE_Average** and **NE_MasT**.

3 Experimental Setup

We used version 2.3.2 of the Indri search engine, developed as part of the Lemur[2] project. While the inference network-based retrieval framework of Indri permits the use of structured queries, the use of language modeling techniques provides better estimates of probabilities for query evaluation. The pseudo-relevance feedback mechanism we used is based on relevance models (Lavrenko and Croft, 2001).

To extract named entities from the queries, we used BBN Identifinder (Bikel et al., 1999). The named entities identified were of type *Person*, *Location*, *Organization*, *Date*, and *Time*.

We used the TREC Robust 2004 and Robust 2005 (Voorhees, 2006) document collections for our experiments. The 2004 Robust collection contains around half a million documents from the Financial Times, the Federal Register, the LA Times, and FBIS. The Robust 2005 collection is the one-million document AQUAINT collection. All the documents were from English newswire. We chose these collections because they and their associated queries are known to be *hard*, and hence present a challenging environment. We stemmed the collections using the Krovetz stemmer provided as part of Indri, and used a manually-created stoplist of twenty terms (*a*, *an*, *and*, *are*, *at*, *as*, *be*, *for*, *in*, *is*, *it*, *of*, *on*, *or*, *that*, *the*, *to*, *was*, *with* and *what*). To determine the best query selection procedure, we analyzed 163 queries from the Robust 2004 track, and used 30 and 50 queries from the 2004 and 2005 Robust tracks respectively for evaluation and user studies.

For all systems, we report mean average precision (MAP) and geometric mean average precision (GMAP). MAP is the most widely used measure in Information Retrieval. While precision is the fraction of the retrieved documents that are relevant, average precision (AP) is a single value obtained by averaging the precision values at each new relevant document observed. MAP is the arithmetic mean of the APs of a set of queries. Similarly, GMAP is the geometric mean of the APs of a set of queries. The GMAP measure is more indicative of performance across an entire set of queries. MAP can be skewed by the presence of a few well-performing queries, and hence is not as good a measure as GMAP from the perspective of measure comprehensive performance.

4 Experiments

We first ran two baseline experiments to record the quality of the available long query and the shorter version. As mentioned in Section 1, we used the description and title sections of each TREC query as surrogates for the long and short versions respectively of a query. The results are presented in the first two rows, Baseline and Pseudo-relevance Feedback (PRF), of Table 2. Measured in terms of MAP and GMAP (Section3), using just the title results in better performance than using the description. This clearly indicates the existence of terms in the description that while elaborating an information need hurt retrieval performance. The result of using pseudo-relevance feedback (PRF) on both the title and description show moderate gains - a known fact about this particular collection and associated train-

[2] `http://www.lemurproject.org`

		MAP	GMAP
Long Query (Description)	Baseline	0.243	0.136
	PRF	0.270	0.124
Short Query (Title)	Baseline	0.249	0.154
	PRF	0.269	0.148
Best sub-query (Combination)	Baseline	0.342	0.270
	PRF	0.343	0.241

Table 2: Results across 163 training queries on the Robust 2004 collection. Using the best sub-query results in almost 50% improvement over the baseline

ing queries.

To show the potential and utility of query re-writing, we first present results that show the upper bound on performance that can obtained by doing so. We ran retrieval experiments with every combination of query terms. For a query of length n, there are 2^n combinations. We limited our experiments to queries of length $n \leq 12$. Selecting the performance obtained by the best sub-query of each query revealed an upper bound in performance almost 50% better than the baseline (Table 2).

To evaluate the automatic sub-query selection procedures developed in Section 2, we performed retrieval experiments using the sub-queries selected using them. The results, which are presented in Table 3, show that the automatic sub-query selection process was a failure. The results of automatic selection were worse than even the baseline, and there was no significant difference between using any of the different sub-query selection procedures.

The failure of the automatic techniques could be attributed to the fact that we were working with the assumption that term co-occurrence could be used to model a user's information need. To see if there was any general utility in using the procedures to select sub-queries, we selected the best-performing sub-query from the top 10 ranked by each selection procedure (Table 4). While the effectiveness in each case as measured by MAP is not close to the best possible MAP, 0.342, they are all significantly better than the baseline of 0.243.

5 Interacting with the user

The final results we presented in the last section hinted at a potential for user interaction. We envisioned providing the user with a list of the top 10 sub-query candidates using a good ranking procedure, and asking her to select the sub-query she felt was most appropriate. This additional round of human intervention could potentially compensate for the inability of the ranking measures to select the best sub-query automatically.

	MAP	GMAP
Baseline	0.243	0.136
Average	0.172	0.025
MaxST	0.172	0.025
NE_Average	0.170	0.023
NE_MaxST	0.182	0.029

Table 3: Score of the highest rank sub-query by various measures.

	MAP	GMAP
Baseline	0.243	0.136
AverageTop10	0.296	0.167
MaxSTTop10	0.293	0.150
NE_AverageTop10	0.278	0.156
NE_MaxSTTop10	0.286	0.159

Table 4: Score of the best sub-query in the top 10 ranked by various measures

5.1 User interface design

We displayed the description (the *long* query) and narrative portion of each TREC query in the interface. The narrative was provided to help the participant understand what information the user who issued the query was interested in. The title was kept hidden to avoid influencing the participant's choice of the best sub-query. A list of candidate sub-queries was displayed along with links that could be clicked on to display a short section of text in a designated area. The intention was to provide an example of what would potentially be retrieved with a high rank if the candidate sub-query were used. The participant used this information to make two decisions - the perceived quality of each sub-query, and the best sub-query from the list. A facility to indicate that none of the candidates were good was also included.

	Percentage of candidates better than baseline
Average	28.5%
MaxST	35.5%
NE_Average	31.1%
NE_MaxST	36.6%

Table 5: Number of candidates from top 10 that exceeded the baseline

5.2 User interface content issues

The two key issues we faced while determining the content of the user interface were:

A. *Deciding which sub-query selection procedure to use to get the top 10 candidate sub-queries:* To determine this in the absence of any significant difference in performance due to the top-ranked candidate selected by each procedure, we looked at the number of candidates each procedure brought into the top 10 that were better than the baseline query, as measured by MAP. This was guided by the belief that greater the number of better candidates in the top 10, the higher the probability that the user would select a better sub-query. Table 5 shows how each of the selection procedures compared. The NE_MaxST ranking procedure had the most number of better sub-queries in the top 10, and hence was chosen.

B. *Displaying context:* Simply displaying a list of 10 candidates without any supportive information would make the task of the user difficult. This was in contrast to query expansion techniques (Anick and Tipirneni, 1999) where displaying a list of terms sufficed as the task of the user was to disambiguate or expand a short query. An experiment was performed in which a single user worked with a set of 30 queries from Robust 2004, and an accompanying set of 10 candidate sub-queries each, twice - once with passages providing context and one with snippets providing context. The top-ranked passage was generated by modifying the candidate query into one that retrieved passages of fixed length instead of documents. Snippets, like those seen along with links to top-ranked documents in the results from almost all popular search engines, were generated after a document-level query was used to query the collection. The order in which the two contexts were presented to the user was randomized to prevent the user from assuming a quality order. We see that presenting the snippet led to better MAP that presenting the passage (Table 6). The reason for this could be that the top-ranking passage we displayed was from a document ranked lower by the document-focussed version of the query. Since we finally measure MAP only with respect to document ranking, and the snippet was generated from the top-ranked document, we hypothesize that this led to the snippet being a better context to display.

	MAP	GMAP
Snippet as Context	0.348	0.170
Passage as Context	0.296	0.151

Table 6: Results showing the MAP over 19 of 30 queries that the user provided selections for using each context type.

6 User Evaluation

We conducted an exploratory study with five participants - four of them were graduate students in computer science while the fifth had a background in the social sciences and was reasonably proficient in the use of computers and internet search engines. The participants worked with 30 queries from Robust 2004, and 50 from Robust 2005[3]. The baseline values reported are automatic runs with the *description* as the query.

Table 7 shows that all five participants[4] were able to choose sub-queries that led to an improvement in performance over the baseline (TREC *title* query only). This improvement is not only on MAP but also on GMAP, indicating that user interaction helped improve a wide spectrum of queries. Most notable were the improvements in P@5 and P@10. This attested to the fact that the interaction technique we explored was precision-enhancing. Another interesting result, from *# sub-queries selected* was that participants were able to decide in a large number of cases that re-writing was either not useful for a query, or that none of the options presented to them were better. Showing context appears to have helped.

[3]Participant 4 looked that only 34 of the 50 queries presented

[4]The p value for testing statistical significance of MAP improvement for Participant 5 was 0.053 - the result very narrowly missed being statistically significant.

	# Queries	# sub-queries selected	% sub-queries better		MAP	GMAP	P@5	p@10
				Baseline	0.203	0.159	0. 476	0.507
1	50	26	80.7%	With Interaction	**0.249**	0.199	0.615	0.580
				Upper Bound	0.336	0.282	0.784	0.719
				Baseline	0.224	0.156	0.484	0.526
2	50	19	78.9%	With Interaction	**0.277**	0.209	0.652	0.621
				Upper Bound	0.359	0.293	0.810	0.742
				Baseline	0.217	0.126	0.452	0.432
3	80	53	73.5%	With Interaction	**0.276**	0.166	0.573	0.501
				Upper Bound	0.354	0.263	0.762	0.654
				Baseline	0.192	0.142	0.462	0.525
4	50(34)	19	68.7%	With Interaction	**0.255**	0.175	0.612	0.600
				Upper Bound	0.344	0.310	0.862	0.800
				Baseline	0.206	0.111	0.433	0.410
5	80	65	61.5%	With Interaction	0.231	0.115	0.486	0.429
				Upper Bound	0.341	0.245	0.738	0.640

Table 7: *# Queries* refers to the number of queries that were presented to the participant while *# sub-queries selected* refers to the number of queries for which the participant chose a sub-query. All scores including upper bounds were calculated only considering the queries for which the participant selected a sub-query. An entry in **bold** means that the improvement in MAP is statistically significant. Statistical significance was measured using a paired t-test, with α set to 0.05.

7 Related Work

Our interest in finding a concise sub-query that effectively captures the information need is reminiscent of previous work in (Buckley et al., 2000). However, the focus was more on balancing the effect of query expansion techniques such that different *concepts* in the query were equally benefited.

Mutual information has been used previously in (Church and Hanks, 1989) to identify collocations of terms for identifying semantic relationships in text. Experiments were confined to bigrams. The use of MaST over a graph of mutual information values to incorporate the most significant dependencies between terms was first noted in (Rijsbergen, 1979). Extensions can be found in a different field - image processing (Kern et al., 2003) - where multivariate mutual information is frequently used.

Work done by (White et al., 2005) provided a basis for our decision to show context for sub-query selection. The useful result that top-ranked sentences could be used to guide users towards relevant material helped us design an user interface that the participants found very convenient to use.

A related problem addressed by (Cronen-Townsend et al., 2002) was determining query quality. This is known to be a very hard problem, and various efforts (Carmel et al., 2006; Vinay et al., 2006) have been made towards formalizing and understanding it.

Previous work (Shapiro and Taksa, 2003) in the web environment attempted to convert a user's natural language query into one suited for use with web search engines. However, the focus was on merging the results from using different sub-queries, and not selection of a single sub-query. Our approach of re-writing queries could be compared to query reformulation, wherein a user follows up a query with successive reformulations of the original. In the web environment, studies have shown that most users still enter only one or two queries, and conduct limited query reformulation (Spink et al., 2002). We hypothesize that the techniques we have developed will be well-suited for search engines like Ask Jeeves where 50% of the queries are in question format

(Spink and Ozmultu, 2002). More experimentation in the Web domain is required to substantiate this.

8 Conclusions

Our results clearly show that shorter reformulations of long queries can greatly impact performance. We believe that our technique has great potential to be used in an adaptive information retrieval environment, where the user starts off with a more general information need and a looser notion of relevance. The initial query can then be made longer to express a most focused information need.

As part of future work, we plan to conduct a more elaborate study with more interaction strategies included. Better techniques to select effective sub-queries are also in the pipeline. Since we used mutual information as the basis for most of our sub-query selection procedures, we could not consider sub-queries that comprised of a single term. We plan to address this issue too in future work.

9 Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsor. We also thank the anonymous reviewers for their valuable comments.

References

James Allan, James P. Callan, W. Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongming Shu. 1996. Inquery at TREC-5. In *TREC*.

Peter G. Anick and Suresh Tipirneni. 1999. The paraphrase search assistant: terminological feedback for iterative information seeking. In *22nd ACM SIGIR Proceedings*, pages 153–159.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. 2000. Using clustering and superconcepts within smart: TREC 6. *Information Processing and Management*, 36(1):109–131.

David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. 2006. What makes a query difficult? In *29th ACM SIGIR Proceedings*, pages 390–397.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *27th ACL Proceedings*, pages 76–83.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms, Second Edition*. The MIT Electrical Engineering and Computer Science Series. The MIT Press.

Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In *25th ACM SIGIR Proceedings*, pages 299–306.

Donna Harman and Chris Buckley. 2004. The NRRC reliable information access (RIA) workshop. In *27th ACM SIGIR Proceedings*, pages 528–529.

Jeffrey P. Kern, Marios Pattichis, and Samuel D. Stearns. 2003. Registration of image cubes using multivariate mutual information. In *Thirty-Seventh Asilomar Conference*, volume 2, pages 1645–1649.

Reiner Kraft, Chi Chao Chang, Farzin Maghoul, and Ravi Kumar. 2006. Searching with context. In *15th International CIKM Conference Proceedings*, pages 477–486.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *24th ACM SIGIR Conference Proceedings*, pages 120–127.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2 edition.

Jacob Shapiro and Isak Taksa. 2003. Constructing web search queries from the user's information need expressed in a natural language. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 1157–1162.

Amanda Spink and H. Cenk Ozmultu. 2002. Characteristics of question format web queries: An exploratory study. *Information Processing and Management*, 38(4):453–471.

Amanda Spink, Bernard J. Jansen, Dietmar Wolfram, and Tefko Saracevic. 2002. From e-sex to e-commerce: Web search changes. *Computer*, 35(3):107–109.

Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *28th ACM SIGIR Proceedings*, pages 449–456.

Vishwa Vinay, Ingemar J. Cox, Natasa Milic-Frayling, and Ken Wood. 2006. On ranking the effectiveness of searches. In *29th ACM SIGIR Proceedings*, pages 398–404.

Ellen M. Voorhees. 2006. The TREC 2005 robust track. *SIGIR Forum*, 40(1):41–48.

Ryen W. White, Joemon M. Jose, and Ian Ruthven. 2005. Using top-ranking sentences to facilitate effective information access: Book reviews. *JAIST*, 56(10):1113–1125.

Combining Outputs from Multiple Machine Translation Systems

Antti-Veikko I. Rosti[a] and **Necip Fazil Ayan**[b] and **Bing Xiang**[a] and
Spyros Matsoukas[a] and **Richard Schwartz**[a] and **Bonnie J. Dorr**[b]
[a]BBN Technologies, 10 Moulton Street, Cambridge, MA 02138
{arosti,bxiang,smatsouk,schwartz}@bbn.com
[b]Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742
{nfa,bonnie}@umiacs.umd.edu

Abstract

Currently there are several approaches to machine translation (MT) based on different paradigms; e.g., phrasal, hierarchical and syntax-based. These three approaches yield similar translation accuracy despite using fairly different levels of linguistic knowledge. The availability of such a variety of systems has led to a growing interest toward finding better translations by combining outputs from multiple systems. This paper describes three different approaches to MT system combination. These combination methods operate on sentence, phrase and word level exploiting information from N-best lists, system scores and target-to-source phrase alignments. The word-level combination provides the most robust gains but the best results on the development test sets (NIST MT05 and the newsgroup portion of GALE 2006 dry-run) were achieved by combining all three methods.

1 Introduction

In recent years, machine translation systems based on new paradigms have emerged. These systems employ more than just the surface-level information used by the state-of-the-art phrase-based translation systems. For example, hierarchical (Chiang, 2005) and syntax-based (Galley et al., 2006) systems have recently improved in both accuracy and scalability. Combined with the latest advances in phrase-based translation systems, it has become more attractive to take advantage of the various outputs in forming consensus translations (Frederking and Nirenburg, 1994; Bangalore et al., 2001; Jayaraman and Lavie, 2005; Matusov et al., 2006).

System combination has been successfully applied in state-of-the-art speech recognition evaluation systems for several years (Fiscus, 1997). Even though the underlying modeling techniques are similar, many systems produce very different outputs with approximately the same accuracy. One of the most successful approaches is consensus network decoding (Mangu et al., 2000) which assumes that the confidence of a word in a certain position is based on the sum of confidences from each system output having the word in that position. This requires aligning the system outputs to form a consensus network and – during decoding – simply finding the highest scoring path through this network. The alignment of speech recognition outputs is fairly straightforward due to the strict constraint in word order. However, machine translation outputs do not have this constraint as the word order may be different between the source and target languages. MT systems employ various re-ordering (distortion) models to take this into account.

Three MT system combination methods are presented in this paper. They operate on the sentence, phrase and word level. The sentence-level combination is based on selecting the best hypothesis out of the merged N-best lists. This method does not generate new hypotheses – unlike the phrase and word-level methods. The phrase-level combination

Proceedings of NAACL HLT 2007, pages 228–235,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

is based on extracting sentence-specific phrase translation tables from system outputs with alignments to source and running a phrasal decoder with this new translation table. This approach is similar to the multi-engine MT framework proposed in (Frederking and Nirenburg, 1994) which is not capable of re-ordering. The word-level combination is based on consensus network decoding. Translation edit rate (TER) (Snover et al., 2006) is used to align the hypotheses and minimum Bayes risk decoding under TER (Sim et al., 2007) is used to select the alignment hypothesis. All combination methods use weights which may be tuned using Powell's method (Brent, 1973) on N-best lists. Both sentence and phrase-level combination methods can generate N-best lists which may also be used as new system outputs in the word-level combination.

Experiments on combining six machine translation system outputs were performed. Three systems were phrasal, two hierarchical and one syntax-based. The systems were evaluated on NIST MT05 and the newsgroup portion of the GALE 2006 dry-run sets. The outputs were evaluated on both TER and BLEU. As the target evaluation metric in the GALE program was human-mediated TER (HTER) (Snover et al., 2006), it was found important to improve both of these automatic metrics.

This paper is organized as follows. Section 2 describes the evaluation metrics and a generic discriminative optimization technique used in tuning of the various system combination weights. Sentence, phrase and word-level system combination methods are presented in Sections 3, 4 and 5. Experimental results on Arabic and Chinese to English newswire and newsgroup test data are presented in Section 6.

2 Evaluation Metrics and Discriminative Tuning

The official metric of the 2006 DARPA GALE evaluation was human-mediated translation edit rate (HTER). HTER is computed as the minimum translation edit rate (TER) between a system output and a targeted reference which preserves the meaning and fluency of the sentence (Snover et al., 2006). The targeted reference is generated by human post-editors who make edits to a reference translation so as to minimize the TER between the reference and the MT output without changing the meaning of the reference. Computing the HTER is very time consuming due to the human post-editing. It is desirable to have an automatic evaluation metric that correlates well with the HTER to allow fast evaluation of the MT systems during development. Correlations of different evaluation metrics have been studied (Snover et al., 2006) but according to various internal HTER experiments it is not clear whether TER or BLEU correlates better. Therefore it is probably safest to try and not degrade either.

The TER of a translation E is computed as

$$\mathrm{TER}(E, E_r) = \frac{\mathrm{Ins} + \mathrm{Del} + \mathrm{Sub} + \mathrm{Shft}}{N_r} \times 100\% \tag{1}$$

where N_r is the total number of words in the reference translation E_r. In the case of multiple references, the edits are counted against all references, N_r is the average number of words in the reference translations and the final TER is computed using the minimum number of edits. The NIST BLEU-4 is a variant of BLEU (Papineni et al., 2002) and is computed as

$$\mathrm{BLEU}(E, E_r) = \exp\Big(\frac{1}{4}\sum_{n=1}^{4}\log p_n(E, E_r)\Big)\gamma(E, E_r) \tag{2}$$

where $p_n(E, E_r)$ is the precision of n-grams in the hypothesis E given the reference E_r and $\gamma(E, E_r) \leq 1$ is a brevity penalty. The n-gram counts from multiple references are accumulated in estimating the precisions.

All system combination methods presented in this paper may be tuned to directly optimize either one of these automatic evaluation metrics. The tuning uses N-best lists of hypotheses with various feature scores. The feature scores may be combined with tunable weights forming an arbitrary scoring function. As the derivatives of this function are not usually available, Brent's modification of Powell's method (Brent, 1973) may be used to find weights that optimize the appropriate evaluation metric in the re-scored N-best list. The optimization starts at a random initial point in the p-dimensional parameter space, first searching through an initial set of basis vectors. As searching repeatedly through the set of basis vectors is inefficient, the direction of the vectors is gradually moved toward a larger positive change in the evaluation metric. To improve the chances of finding a global optimum, the algorithm is repeated with varying initial values. The modified Powell's method has been previously used in optimizing the weights of a standard feature-based MT decoder in (Och, 2003) where a more efficient algorithm for log-linear models was proposed. However, this is specific to log-linear models and cannot be easily extended for more complicated functions.

3 Sentence-Level Combination

The first combination method is based on re-ranking a merged N-best list. A confidence score from each system is assigned to each unique hypothesis in the merged list. The confidence scores for each hypothesis are used to produce a single score which, combined with a 5-gram language model score, determines a new ranking of the hypotheses.

3.1 Hypothesis Confidence Estimation

Generalized linear models (GLMs) have been applied for confidence estimation in speech recognition (Siu and Gish, 1999). The logit model, which models the log odds of an event as a linear function of the features, can be used in confidence estimation. The confidence P_{ij} for a system i generating a hypothesis j may be modeled as

$$\log \frac{P_{ij}}{1 - P_{ij}} = \sum_{l=1}^{L} w_{il} x_{ijl} \tag{3}$$

where each system has L weights w_{il}, and x_{ijl} is the lth feature for system i and hypothesis j. The features used in this work were:

1. Rank in the system's N-best list;
2. Sentence posterior with system-specific total score scaling factors;
3. System's total score;
4. Number of words in the hypothesis;
5. System-specific bias.

If the system i did not generate the hypothesis j, the confidence is set to zero. To prevent overflow in exponentiating the summation in Equation 3, the features have to be scaled. In the experiments, feature scaling factors were estimated from the tuning data to limit the feature values between $[0, 1]$. The same scaling factors have to be applied to the features obtained from the test data.

The total confidence score of hypothesis j is obtained from the system confidences P_{ij} as

$$P_j = \alpha \frac{N_j}{N_s} + \beta \frac{1}{N_s} \sum_{i=1}^{N_s} P_{ij} + \gamma \max_i P_{ij} + \delta \frac{1}{N_j} \sum_{i=1}^{N_s} P_{ij} \tag{4}$$

where N_j is the number of systems generating the hypothesis j (i.e., the number of non-zero P_{ij} for j) and N_s is the number of systems. The weights α through δ are constrained to sum to one; i.e., there are three free parameters. These weights can balance the total confidence between the number of systems generating the hypothesis (votes), and the sum, maximum and average of the system confidences.

3.2 Sentence Posterior Estimation

The second feature in the GLM is the sentence posterior estimated from the N-best list. A sentence posterior may simply be estimated from an N-best list by scaling the system scores for all hypotheses to sum to one. When combining several systems based on different translation paradigms and feature sets, the system scores may not be comparable. The total scores may be scaled to obtain more consistent sentence posteriors. The scaled posterior estimated from an N-best list may be written as

$$P_{ij} = \exp\Big(s_i L_{ij} - \log\Big(\sum_{k=1}^{N} \exp(s_i L_{ik})\Big)\Big) \tag{5}$$

where s_i is the scaling factor for system i and L_{ij} is the log-score system i assigns to hypothesis j. The scaling factors may be tuned to optimize the evaluation metric in the same fashion as the logit model weights in Section 3.1. Equation 4 may be used to assign total posteriors for each unique hypothesis and the weights may be tuned using Powell's method on N-best lists as described in Section 2.

3.3 Hypothesis Re-ranking

The hypothesis confidence may be log-linearly combined with a 5-gram language model (LM) score to yield the final score as follows

$$L_j = \log P_j + \nu L_j^{5gr} + \mu W_j \tag{6}$$

where W_j is the number of words in hypothesis j. The number of words is commonly used in LM re-scoring to balance the LM scores between hypotheses of different lengths. The number of free parameters in the sentence-level combination method is given by $N_s + N_s L + 8$ where N_s is the number of systems and L is the number of features; i.e., N_s system score scaling factors (s_i), three free interpolation weights (Equation 4) for the scaling factor estimation, $N_s L$ GLM weights (w_{il}), three free interpolation weights (Equation 4) for the hypothesis confidence estimation and two free LM re-scoring weights (Equation 6). All parameters may be tuned using Powell's method on N-best lists as described in Section 2.

The tuning of the sentence-level combination method may be summarized as follows:

1. Merge individual N-best lists to form a large N-best list with unique hypotheses;
2. Estimate total score scaling factors as described in Section 3.2;
3. Collect GLM feature scores for each unique hypothesis;
4. Estimate GLM feature scaling factors as described in Section 3.1;
5. Scale the GLM features;
6. Estimate GLM weights, combination weights and LM re-scoring weights as described above;
7. Re-rank the merged N-best list using the new weights.

Testing the sentence-level combination has the same steps as the tuning apart from all estimation steps; i.e., steps 1, 3, 5 and 7.

4 Phrase-Level Combination

The phrase-level combination is based on extracting a new phrase translation table from each system's target-to-source phrase alignments and re-decoding the source sentence using this new translation table and a language model. In this work, the target-to-source phrase alignments were available from the individual systems. If the alignments are not available, they can be automatically generated; e.g., using GIZA++ (Och and Ney, 2003). The phrase translation table is generated for each source sentence using confidence scores derived from sentence posteriors with system-specific total score scaling factors and similarity scores based on the agreement among the phrases from all systems.

4.1 Phrase Confidence Estimation

Each phrase has an initial confidence based on the sentence posterior P_{ij} estimated from an N-best list in the same fashion as in Section 3.2. The confidence of the phrase table entry is increased if several systems agree on the target words. The agreement is measured by four levels of similarity:

1. Same source interval, same target words, and same original distortion;
2. Same source interval, same target words, with different original distortion;
3. Overlapping source intervals with the same target words;
4. Overlapping target words.

S_{ilm} represents the similarity of a given phrase m to all the hypotheses in the system i at the similarity level l. Basically, if there is a similar phrase in a given hypothesis j in the system i to the phrase m, the similarity score S_{ilm} is increased by P_{ij}. Note that each phrase in one hypothesis is similar to another hypothesis at only one similarity level, so one hypothesis can contribute to S_{ilm} at only one similarity level. The final confidence of the phrase table entry is defined as

$$\begin{aligned} P_m = \log \Big(& \alpha \sum_{i,l} w_i v_l S_{ilm} \\ & + \beta \frac{1}{\sum_{i,l:S_{ilm}\neq 0} w_i v_l} \sum_{i,l:S_{ilm}\neq 0} w_i v_l S_{ilm} \\ & + \gamma \max_i \sum_l w_i v_l S_{ilm} \Big) \end{aligned} \tag{7}$$

where w_i are system weights and v_l are similarity score weights. The parameters α through γ interpolate between the sum, average and maximum of the similarity scores. These interpolation weights and the system weights w_i are constrained to sum to one. The number of tunable combination weights, in addition to normal decoder weights, is $N_s + N_l + 1$ where N_s is the number of systems and N_l is the number of similarity levels; i.e., $N_s - 1$ free system weights, N_l similarity score weights and two free interpolation weights.

4.2 Phrase-Based Decoding

The phrasal decoder used in the phrase-level combination is based on standard beam search (Koehn, 2004). The decoder features are: a trigram language model score, number of target phrases, number of target words, phrase distortion, phrase distortion computed over the original translations and phrase translation confidences estimated in Section 4.1. The total score for a hypothesis is computed as a log-linear combination of these features. The feature weights and combination weights (system and similarity) may be tuned using Powell's method on N-best lists as described in Section 2.

The phrase-level combination tuning can be summarized as follows:

1. Estimate sentence posteriors given the total score scaling factors;
2. Collect all M unique phrase table entries from each hypothesis accumulating the similarity scores S_{ilm};
3. Combine the similarity scores to form phrase confidences according to Equation 7;
4. Decode the source sentences using the current weights to generate an N-best list;
5. Estimate new decoder and combination weights as described above.

Testing the phrase-level combination is performed by following steps 1 through 4.

5 Word-Level Combination

The third combination method is based on confusion network decoding. In confusion network decoding, the words in all hypotheses are aligned against each other to form a graph with word alternatives (including nulls) for each alignment position. Each aligned word is assigned a score relative to the votes or word confidence scores (Fiscus, 1997; Mangu et al., 2000) derived from the hypotheses. The decoding is carried out by picking the words with the highest scores along the graph. In speech recognition, this results in minimum expected word error rate (WER) hypothesis (Mangu et al., 2000) or equivalently minimum Bayes risk (MBR) under WER with uniform target sentence posterior distribution (Sim et al., 2007).

In machine translation, aligning hypotheses is more complicated compared to speech recognition since the target words do not necessarily appear in the same order. So far, confusion networks have been applied in MT system combination using three different alignment procedures: WER (Bangalore et al., 2001), GIZA++ alignments (Matusov et al., 2006) and TER (Sim et al., 2007). WER alignments do not allow shifts, GIZA++ alignments require careful training and are not always reliable. TER alignments do not guarantee that similar but lexically different words are aligned correctly but TER does not require training new models and allows shifts (Snover et al., 2006). This work extends the approach proposed in (Sim et al., 2007).

5.1 Confusion Network Generation

Due to the varying word order in the MT hypotheses, the decision of confusion network *skeleton* is essential. The skeleton determines the general word order of the combined hypothesis. One option would be to use the output from the system with the best performance on some development set. However, it was found that this approach did not always yield better combination output compared to the best single system on all evaluation metrics. Instead of using a single system output as the skeleton, the hypothesis that best agrees with the other hypotheses on average may be used. In this paper, the minimum average TER score of one hypothesis against all other hypotheses was used as follows

$$E_r = \arg\min_i \sum_{j=1}^{N_s} \mathrm{TER}(E_j, E_i) \tag{8}$$

This may be viewed as the MBR hypothesis under TER given uniform target sentence posterior distribution (Sim et al., 2007). It is also possible to compute the MBR hypothesis under BLEU.

Finding the MBR hypothesis requires computing the TER against all hypotheses to be aligned. It was found that aligning more than one hypothesis ($N = 10$) from each system to the skeleton improves the combination outputs. However, only the rank-1 hypotheses were considered as skeletons due to the complexity of the TER alignment. The confidence score assigned to each word was chosen to be $1/(1 + rank)$ where the $rank$ was based on the rank of the aligned hypothesis in the system's N-best. This was found to yield better scores than simple votes.

5.2 Tunable System Weights

The word-level combination method described so far does not require any tuning. To allow a variety of outputs with different degrees of confidence to be combined, system weights may be used. A confusion network may be represented as a standard word lattice with all paths traveling via all nodes. The links in this lattice represent the alternative words (including nulls) at the corresponding position in the string. Confusion network decoding may be viewed as finding the highest scoring path through this lattice with summing all word scores along the path. The standard lattice decoding algorithms may also be used to generate N-best lists from the confusion network. The simplest way to introduce system weights is to accumulate system-specific scores along the paths and combine these scores linearly with the weights. The system weights may be tuned using Powell's method on N-best lists as described in Section 2.

The word-level combination tuning can be summarized as follows:

1. Extract 10-best lists from the MT outputs;
2. Align each 10-best against each rank-1 hypothesis using TER;
3. Choose the skeleton (Equation 8);
4. Generate a confusion network lattice with the current system weights;
5. Generate N-best list hypothesis and score files from the lattice;
6. Estimate system weights as described above;

Arabic	Newswire		Newsgroups	
	TER	BLEU	TER	BLEU
system A	42.98	49.58	59.73	20.36
system B	43.79	47.06	61.55	18.08
system C	43.92	47.87	60.81	18.08
system D	40.75	52.09	59.25	20.28
system E	42.19	50.86	59.85	19.73
system F	44.30	50.15	61.74	20.61
phrcomb	40.45	53.70	59.90	21.49
sentcomb	41.56	52.18	60.21	19.77
no weights 6	39.33	53.66	58.15	20.61
TER 6	39.41	54.37	58.21	20.85
TER 8	39.43	54.40	57.96	21.44

Table 1: Mixed-case TER and BLEU scores on Arabic NIST MT05 (newswire) and the newsgroups portion of the GALE 2006 dry-run data.

7. Re-rank the N-best list using the new weights.

Testing the word-level combination has the same steps as the tuning apart from steps 6 and 7.

6 Experiments

Six systems trained on all data available for GALE 2006 evaluation were used in the experiments to demonstrate the performance of all three system combination methods on Arabic and Chinese to English MT tasks. Three systems were phrase-based (A, C and E), two hierarchical (B and D) and one syntax-based (F). The phrase-based systems used different sets of features and re-ordering approaches. The hierarchical systems used different rule sets. All systems were tuned on NIST MT02 evaluation sets with four references. Systems A and B were tuned to minimize TER, the other systems were tuned to maximize BLEU.

As discussed in Section 2, the system combination tuning metric was chosen so that gains were observed in both TER and BLEU on development test sets. NIST MT05 comprising only newswire data (1056 Arabic and 1082 Chinese sentences) with four reference translations and the newsgroup portion of the GALE 2006 dry-run (203 Arabic and 126 Chinese sentences) with one reference translation were used as the test sets. It was found that minimizing TER on Arabic also resulted in higher BLEU scores compared to the best single system. However,

Chinese	Newswire		Newsgroups	
	TER	BLEU	TER	BLEU
system A	56.57	29.63	68.61	13.20
system B	56.30	29.62	69.87	12.33
system C	59.48	31.32	69.37	13.91
system D	58.32	33.77	67.61	16.86
system E	58.46	32.40	69.08	15.08
system F	56.79	35.30	68.08	16.31
phrcomb	56.50	35.33	68.48	15.88
sentcomb	56.71	36.24	69.50	16.11
no weights 6	53.80	36.17	66.87	15.90
BLEU 6	54.34	36.44	66.50	16.44
BLEU 8	54.86	36.90	66.45	17.32

Table 2: Mixed-case TER and BLEU scores on Chinese NIST MT05 (newswire) and the newsgroups portion of the GALE 2006 dry-run data.

minimizing TER on Chinese resulted in significantly lower BLEU. So, TER was used in tuning the combination weights on Arabic and BLEU on Chinese.

The sentence and phrase-level combination weights were tuned on NIST MT03 evaluation sets. On the tuning sets, both methods yield about 0.5%-1.0% gain in TER and BLEU. The mixed-case TER and BLEU scores on both test sets are shown in Table 1 for Arabic and Table 2 for Chinese (`phrcomb` represents phrase and `sentcomb` sentence-level combination). The phrase-level combination seems to outperform the sentence-level combination in terms of both metrics on Arabic although gains over the best single system are modest, if any. On Chinese, the sentence-level combination yields higher BLEU scores than the phrase-level combination. The combination BLEU scores on the newsgroup data are not higher than the best system, though.

The word-level combination was evaluated in three settings. First, simple confusion network decoding with six systems without system weights was performed (`no weights 6` in the tables). Second, system weights were trained for combining six systems (`TER/BLEU 6` in the tables). Finally, all six system outputs as well as the sentence and phrase-level combination outputs were combined with system weights (`TER/BLEU 8` in the tables). The 6-way combination weights were tuned on merged NIST MT03 and MT04 evaluation sets and the 8-way combination weights were tuned only on NIST MT04 since the sentence and phrase-level combination methods were already tuned on NIST MT03. The word-level combination yields about 2.0%-3.0% gain in TER and 2.0%-4.0% gain in BLEU on the tuning sets. The test set results show that the simple confusion network decoding without system weights yields very good scores, mostly better than either sentence or phrase-level combination. The system weights seem to yield even higher BLEU scores but not always lower TER scores on both languages. Despite slightly hurting the TER score on Arabic, the `TER 8` combination result was considered the best due to the highest BLEU and significantly lower TER compared to any single system. Similarly, the `BLEU 8` was considered the best combination result on Chinese. Internal HTER experiments showed that `BLEU 8` yielded lower scores after post-editing even though `no weights 6` had lower automatic TER score.

7 Conclusions

Three methods for machine translation system combination were presented in this paper. The sentence-level combination was based on re-ranking a merged N-best list using generalized linear models with features derived from each system's output. The combination yields slight gains on the tuning set. However, the gains were very small, if any, on the test sets. The re-ranked N-best lists were used successfully in the word-level combination method as new system outputs. Various other features may be explored in this framework although the tuning may be limited by the chosen optimization method in the higher dimensional parameter space.

The phrase-level combination was based on deriving a new phrase translation table from the alignments to source provided in all system outputs. The phrase translation scores were based on the level of agreement between the system outputs and sentence posterior estimates. A standard phrasal decoder with the new phrase table was used to produce the final combination output. The handling of the alignments from non-phrasal decoders may not be optimal, though. The phrase-level combination yields fairly good gains on the tuning sets. However, the performance does not seem to generalize to the test sets used in this work. As usual, the phrasal decoder can generate N-best lists which were used successfully in the word-level combination method as new system outputs.

The word-level combination method based on consensus network decoding seems to be very robust and yield good gains over the best single system even without any tunable weights. The decision of the skeleton is crucial. Minimum Bayes Risk decoding under translation edit rate was used to select the skeleton. Compared to the best possible skeleton decision – according to an oracle experiment – further gains might be obtained by using better decision approach. Also, the alignment may be improved by taking the target-to-source alignments into account and allowing synonyms to align. The confusion network decoding at the word level does not necessarily retain coherent phrases as no language model constraints are taken into account. LM re-scoring might alleviate this problem.

This paper has provided evidence that outputs from six very different MT systems, tuned for two different evaluation metrics, may be combined to yield better outputs in terms of different evaluation metrics. The focus of the future work will be to address the individual issues in the combination methods mentioned above. It would also be interesting to investigate how much different systems contribute to the overall gain achieved via system combination.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program (approved for public release, distribution unlimited). The authors would like to thank ISI and University of Edinburgh for sharing their MT system outputs.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proc. ASRU*, pages 351–354.

Richard P. Brent. 1973. *Algorithms for Minimization Without Derivatives*. Prentice-Hall.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proc. ASRU*, pages 347–354.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proc. ANLP*, pages 95–100.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proc. COLING/ACL*, pages 961–968.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proc. EAMT*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proc. AMTA*.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400.

Evgeny Matusov, Nicola Ueffi ng, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. EACL*, pages 33–40.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. 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 *Proc. ACL*, pages 311–318.

Khe Chai Sim, William J. Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proc. ICASSP*.

Manhung Siu and Herbert Gish. 1999. Evaluation of word confi dence for speech recognition systems. *Computer Speech and Language*, 13(4):299–319.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciula, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. AMTA*.

Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming

Pascal Denis and **Jason Baldridge**
Department of Linguistics
University of Texas at Austin
{denis,jbaldrid}@mail.utexas.edu

Abstract

Standard pairwise coreference resolution systems are subject to errors resulting from their performing anaphora identification as an implicit part of coreference resolution. In this paper, we propose an integer linear programming (ILP) formulation for coreference resolution which models anaphoricity and coreference as a joint task, such that each local model informs the other for the final assignments. This joint ILP formulation provides f-score improvements of 3.7-5.3% over a base coreference classifier on the ACE datasets.

1 Introduction

The task of coreference resolution involves imposing a partition on a set of entity mentions in a document, where each partition corresponds to some entity in an underlying discourse model. Most work treats coreference resolution as a binary classification task in which each decision is made in a pairwise fashion, independently of the others (McCarthy and Lehnert, 1995; Soon et al., 2001; Ng and Cardie, 2002b; Morton, 2000; Kehler et al., 2004).

There are two major drawbacks with most systems that make pairwise coreference decisions. The first is that identification of anaphora is done *implicitly* as part of the coreference resolution. Two common types of errors with these systems are cases where: (i) the system mistakenly identifies an antecedent for non-anaphoric mentions, and (ii) the system does not try to resolve an actual anaphoric mention. To reduce such errors, Ng and Cardie (2002a) and Ng (2004) use an *anaphoricity* classifier –which has the sole task of saying whether or not *any* antecedents should be identified for each mention– as a filter for their coreference system. They achieve higher performance by doing so; however, their setup uses the two classifiers in a cascade. This requires careful determination of an anaphoricity threshold in order to not remove too many mentions from consideration (Ng, 2004). This sensitivity is unsurprising, given that the tasks are codependent.

The second problem is that most coreference systems make each decision independently of previous ones in a greedy fashion (McCallum and Wellner, 2004). Clearly, the determination of membership of a particular mention into a partition should be conditioned on how well it matches the entity as a whole. Since independence between decisions is an unwarranted assumption for the task, models that consider a more global context are likely to be more appropriate. Recent work has examined such models; Luo et al. (2004) using Bell trees, and McCallum and Wellner (2004) using conditional random fields, and Ng (2005) using rerankers.

In this paper, we propose to recast the task of coreference resolution as an optimization problem, namely an integer linear programming (ILP) problem. This framework has several properties that make it highly suitable for addressing the two aforementioned problems. The first is that it can utilize existing classifiers; ILP performs global inference based on their output rather than formulating a new inference procedure for solving the basic task. Second, the ILP approach supports inference over multiple classifiers, without having to fiddle with special parameterization. Third, it is much more efficient than conditional random fields, especially when long-distance features are utilized (Roth and Yih, 2005). Finally, it is straightforward to create categorical global constraints with ILP; this is done in a declarative manner using inequalities on the assignments to indicator variables.

This paper focuses on the first problem, and proposes to model anaphoricity determination and coreference resolution as a joint task, wherein the decisions made by each locally trained model are mutually constrained. The presentation of the ILP model proceeds in two steps. In the first, intermediary step, we simply use ILP to find a global assignment based on decisions made by the coreference classifier alone. The resulting assignment is one that maximally agrees with the decisions of the classifier, that is, where *all and only* the links predicted to be coreferential are used for constructing the chains. This is in contrast with the usual clustering algorithms, in which a *unique* antecedent is typically picked for each anaphor (e.g., the most probable or the most recent). The second step provides the joint formulation: the coreference classifier is now combined with an anaphoricity classifier and constraints are added to ensure that the ultimate coreference and anaphoricity decisions are mutually consistent. Both of these formulations achieve significant performance gains over the base classifier. Specifically, the joint model achieves f-score improvements of 3.7-5.3% on three datasets.

We begin by presenting the basic coreference classifier and anaphoricity classifier and their performance, including an upperbound that shows the limitation of using them in a cascade. We then give the details of our ILP formulations and evaluate their performance with respect to each other and the base classifier.

2 Base models: coreference classifier

The classification approach tackles coreference in two steps by: (i) estimating the probability, $P_C(\text{COREF}|\langle i,j\rangle)$, of having a coreferential outcome given a pair of mentions $\langle i,j\rangle$, and (ii) applying a selection algorithm that will single out a unique candidate out of the subset of candidates i for which the probability $P_C(\text{COREF}|\langle i,j\rangle)$ reaches a particular value (typically .5).

We use a maximum entropy model for the coreference classifier. Such models are well-suited for coreference, because they are able to handle many different, potentially overlapping learning features without making independence assumptions. Previous work on coreference using maximum entropy includes (Kehler, 1997; Morton, 1999; Morton, 2000). The model is defined in a standard fashion as follows:

$$P_C(\text{COREF}|\langle i,j\rangle) = \frac{exp(\sum_{k=1}^{n} \lambda_k f_k(\langle i,j\rangle, \text{COREF}))}{Z(\langle i,j\rangle)} \tag{1}$$

$Z(\langle i,j\rangle)$ is a normalization factor over both outcomes (COREF and $\neg$COREF). Model parameters are estimated using maximum entropy (Berger et al., 1996). Specifically, we estimate parameters with the limited memory variable metric algorithm implemented in the Toolkit for Advanced Discriminative Modeling[1] (Malouf, 2002). We use a Gaussian prior with a variance of 1000 — no attempt was made to optimize this value.

Training instances for the coreference classifier are constructed based on pairs of mentions of the form $\langle i,j\rangle$, where j and i are the descriptions for an anaphor and one of its candidate antecedents, respectively. Each such pair is assigned either a label COREF (i.e. a positive instance) or a label $\neg$COREF (i.e. a negative instance) depending on whether or not the two mentions corefer. In generating the training data, we followed the method of (Soon et al., 2001) creating for each anaphor: (i) a *positive instance* for the pair $\langle i,j\rangle$ where i is the closest antecedent for j, and (ii) a *negative instance* for each pair $\langle i,k\rangle$ where k intervenes between i and j.

Once trained, the classifier is used to create a set of coreferential links for each test document; these links in turn define a partition over the entire set of mentions. In the system of Soon et. al. (2001) system, this is done by pairing each mention j with each preceding mention i. Each test instance $\langle i,j\rangle$ thus

[1]Available from `tadm.sf.net`.

formed is then evaluated by the classifier, which returns a probability representing the likelihood that these two mentions are coreferential. Soon et. al. (2001) use "Closest-First" selection: that is, the process terminates as soon as an antecedent (i.e., a test instance with probability $> .5$) is found or the beginning of the text is reached. Another option is to pick the antecedent with the best overall probability (Ng and Cardie, 2002b).

Our features for the coreference classifier fall into three main categories: (i) features of the anaphor, (ii) features of antecedent mention, and (iii) relational features (i.e., features that describe properties which hold between the two mentions, e.g. distance). This feature set is similar (though not equivalent) to that used by Ng and Cardie (2002a). We omit details here for the sake of brevity — the ILP systems we employ here could be equally well applied to many different base classifiers using many different feature sets.

3 Base models: anaphoricity classifier

As mentioned in the introduction, coreference classifiers such as that presented in section 2 suffer from errors in which (a) they assign an antecedent to a non-anaphor mention or (b) they assign no antecedents to an anaphoric mention. Ng and Cardie (2002a) suggest overcoming such failings by augmenting their coreference classifier with an anaphoricity classifier which acts as a filter during model usage. Only the mentions that are deemed anaphoric are considered for coreference resolution. Interestingly, they find a degredation in performance. In particular, they obtain significant improvements in precision, but with larger losses in recall (especially for proper names and common nouns). To counteract this, they add *ad hoc* constraints based on string matching and extended mention matching which force certain mentions to be resolved as anaphors regardless of the anaphoricity classifier. This allows them to improve overall f-scores by 1-3%. Ng (2004) obtains f-score improvements of 2.8-4.5% by tuning the anaphoricity threshold on held-out data.

The task for the anaphoricity determination component is the following: one wants to decide for each mention i in a document whether i is anaphoric or not. That is, this task can be performed using a simple binary classifier with two outcomes: ANAPH and ¬ANAPH. The classifier estimates the conditional probabilities $P(\text{ANAPH}|i)$ and predicts ANAPH for i when $P(\text{ANAPH}|i) > .5$.

We use the following model for our anaphoricity classifier:

$$P_A(\text{ANAPH}|i) = \frac{exp(\sum_{k=1}^{n} \lambda_k f_k(i, \text{ANAPH}))}{Z(i)} \qquad (2)$$

This model is trained in the same manner as the coreference classifier, also with a Gaussian prior with a variance of 1000.

The features used for the anaphoricity classifier are quite simple. They include information regarding (1) the mention itself, such as the number of words and whether it is a pronoun, and (2) properties of the potential antecedent set, such as the number of preceding mentions and whether there is a previous mention with a matching string.

4 Base model results

This section provides the performance of the pairwise coreference classifier, both when used alone (COREF-PAIRWISE) and when used in a cascade where the anaphoricity classifier acts as a filter on which mentions should be resolved (AC-CASCADE). In both systems, antecedents are determined in the manner described in section 2.

To demonstrate the inherent limitations of cascading, we also give results for an oracle system, ORACLE-LINK, which assumes *perfect linkage*. That is, it always picks the correct antecedent for an anaphor. Its only errors are due to being unable to resolve mentions which were marked as non-anaphoric (by the imperfect anaphoricity classifier) when in fact they were anaphoric.

We evaluate these systems on the datasets from the ACE corpus (Phase 2). This corpus is divided into three parts, each corresponding to a different genre: newspaper texts (NPAPER), newswire texts (NWIRE), and broadcasted news transcripts (BNEWS). Each of these is split into a `train` part and a `devtest` part. Progress during the development phase was determined by using cross-validation on only the training set for the NPAPER

System	BNEWS			NPAPER			NWIRE		
	R	P	F	R	P	F	R	P	F
COREF-PAIRWISE	54.4	77.4	63.9	58.1	80.7	67.6	53.8	78.2	63.8
AC-CASCADE	51.1	79.7	62.3	53.7	79.0	63.9	53.0	81.8	64.3
ORACLE-LINK	69.4	100	82.0	71.2	100	83.1	66.7	100	80.0

Table 1: Recall (R), precision (P), and f-score (F) on the three ACE datasets for the basic coreference system (COREF-PAIRWISE), the anaphoricity-coreference cascade system (AC-CASCADE), and the oracle which performs perfect linkage (ORACLE-LINK). The first two systems make strictly local pairwise coreference decisions.

section. No human-annotated linguistic information is used in the input. The corpus text was preprocessed with the OpenNLP Toolkit[2] (i.e., a sentence detector, a tokenizer, a POS tagger, and a Named Entity Recognizer).

In our experiments, we consider only the *true* ACE mentions. This is because our focus is on evaluating pairwise local approaches versus the global ILP approach rather than on building a full coreference resolution system. It is worth noting that previous work tends to be vague in both these respects: details on mention filtering or providing performance figures for markable identification are rarely given.

Following common practice, results are given in terms of recall and precision according to the standard model-theoretic metric (Vilain et al., 1995). This method operates by comparing the equivalence classes defined by the resolutions produced by the system with the gold standard classes: these are the two "models". Roughly, the scores are obtained by determining the minimal perturbations brought to one model in order to map it onto the other model. Recall is computed by trying to map the predicted chains onto the true chains, while precision is computed the other way around. We test significant differences with paired t-tests ($p < .05$).

The anaphoricity classifier has an average accuracy of 80.2% on the three ACE datasets (using a threshold of $.5$). This score is slightly lower than the scores reported by Ng and Cardie (2002a) for another data set (MUC).

Table 1 summarizes the results, in terms of recall (R), precision (P), and f-score (F) on the three ACE data sets. As can be seen, the AC-CASCADE system generally provides slightly better precision at the expense of recall than the COREF-PAIRWISE system, but the performance varies across the three datasets. The source of this variance is likely due to the fact that we applied a uniform anaphoricity threshold of .5 across all datasets; Ng (2004) optimizes this threshold for each of the datasets: .3 for BNEWS and NWIRE and .35 for NPAPER. This variance reinforces our argument for determining anaphoricity and coreference jointly.

The limitations of the cascade approach are also shown by the oracle results. Even if we had a system that can pick the correct antecedents for all truly anaphoric mentions, it would have a maximum recall of roughly 70% for the different datasets.

5 Integer programming formulations

The results in the previous section demonstrate the limitations of a cascading approach for determining anaphoricity and coreference with separate models. The other thing to note is that the results in general provide a lot of room for improvement — this is true for other state-of-the-art systems as well. The integer programming formulation we provide here has qualities which address both of these issues. In particular, we define two objective functions for coreference resolution to be optimized with ILP. The first uses only information from the coreference classifier (COREF-ILP) and the second integrates both anaphoricity and coreference in a joint formulation (JOINT-ILP). Our problem formulation and use of ILP are based on both (Roth and Yih, 2004) and (Barzilay and Lapata, 2006).

For solving the ILP problem, we use lp_solve, an open-source linear programming solver which implements the simplex and the Branch-and-Bound

[2]Available from `opennlp.sf.net`.

methods.[3] In practice, each test document is processed to define a distinct ILP problem that is then submitted to the solver.

5.1 COREF-ILP: coreference-only formulation

Barzilay and Lapata (2006) use ILP for the problem of aggregation in natural language generation: clustering sets of propositions together to create more concise texts. They cast it as a set partitioning problem. This is very much like coreference, where each partition corresponds to an entity in a discourse model.

COREF-ILP uses an objective function that is based on *only* the coreference classifier and the probabilities it produces. Given that the classifier produces probabilities $p_C = P_C(\text{COREF}|i,j)$, the assignment cost of commiting to a coreference link is $c^C_{\langle i,j\rangle} = -log(p_C)$. A complement assignment cost $\overline{c}^C_{\langle i,j\rangle} = -log(1-p_C)$ is associated with choosing not to establish a link. In what follows, M denotes the set of mentions in the document, and P the set of possible coreference links over these mentions (i.e., $P = \{\langle i,j\rangle | \langle i,j\rangle \in M \times M$ and $i < j\}$). Finally, we use indicator variables $x_{\langle i,j\rangle}$ that are set to 1 if mentions i and j are coreferent, and 0 otherwise. The objective function takes the following form:

$$min \sum_{\langle i,j\rangle \in P} c^C_{\langle i,j\rangle} \cdot x_{\langle i,j\rangle} + \overline{c}^C_{\langle i,j\rangle} \cdot (1 - x_{\langle i,j\rangle}) \quad (3)$$

subject to:

$$x_{\langle i,j\rangle} \in \{0,1\} \qquad \forall \langle i,j\rangle \in P \quad (4)$$

This is essentially identical to Barzilay and Lapata's objective function, except that we consider only pairs in which the i precedes the j (due to the structure of the problem). Also, we minimize rather than maximize due to the fact we transform the model probabilities with $-log$ (like Roth and Yih (2004)).

This preliminary objective function merely guarantees that ILP will find a global assignment that maximally agrees with the decisions made by the coreference classifier. Concretely, this amounts to taking all (and only) those links for which the classifier returns a probability above .5. This formulation does not yet take advantage of information from a classifier that specializes in anaphoricity; this is the subject of the next section.

[3]Available from `http://lpsolve.sourceforge.net/`.

5.2 JOINT-ILP: joint anaphoricity-coreference formulation

Roth and Yih (2004) use ILP to deal with the joint inference problem of named entity and relation identification. This requires labeling a set of named entities in a text with labels such as *person* and *location*, and identifying relations between them such as *spouse_of* and *work_for*. In theory, each of these tasks would likely benefit from utilizing the information produced by the other, but if done as a cascade will be subject to propogation of errors. Roth and Yih thus set this up as problem in which each task is performed separately; their output is used to assign costs associated with indicator variables in an objective function, which is then minimized subject to constraints that relate the two kinds of outputs. These constraints express qualities of what a global assignment of values for these tasks must respect, such as the fact that the arguments to the *spouse_of* relation must be entities with *person* labels. Importantly, the ILP objective function encodes not only the best label produced by each classifier for each decision; it utilizes the probabilities (or scores) assigned to each label and attempts to find a global optimum (subject to the constraints).

The parallels to our anaphoricity/coreference scenario are straightforward. The anaphoricity problem is like the problem of identifying the type of entity (where the labels are now ANAPH and ¬ANAPH), and the coreference problem is like that of determining the relations between mentions (where the labels are now COREF or ¬COREF).

Based on these parallels, the JOINT-ILP system brings the two decisions of anaphoricity and coreference together by including both in a single objective function and including constraints that ensure the *consistency* of a solution for both tasks. Let c^A_j and $\overline{c}^A_j$ be defined analogously to the coreference classifier costs for $p_A = P_A(\text{ANAPH}|j)$, the probability the anaphoricity classifier assigns to a mention j being anaphoric. Also, we have indicator variables y_j that are set to 1 if mention j is anaphoric and 0 otherwise. The objective function takes the following form:

$$min \sum_{\langle i,j\rangle \in P} c^C_{\langle i,j\rangle} \cdot x_{\langle i,j\rangle} + \overline{c}^C_{\langle i,j\rangle} \cdot (1 - x_{\langle i,j\rangle}) + \sum_{j \in M} c^A_j \cdot y_j + \overline{c}^A_j \cdot (1 - y_j) \quad (5)$$

subject to:

$$x_{\langle i,j\rangle} \in \{0,1\} \qquad \forall \langle i,j\rangle \in P \quad (6)$$

$$y_j \in \{0,1\} \qquad \forall j \in M \quad (7)$$

The structure of this objective function is very similar to Roth and Yih's, except that we do not utilize constraint costs in the objective function itself. Roth and Yih use these to make certain combinations impossible (like a *location* being an argument to a *spouse_of* relation); we enforce such effects in the constraint equations instead.

The joint objective function (5) does not constrain the assignment of the $x_{\langle i,j\rangle}$ and y_j variables to be consistent with one another. To enforce consistency, we add further constraints. In what follows, M_j is the set of all mentions preceding mention j in the document.

Resolve only anaphors: if a pair of mentions $\langle i, j\rangle$ is coreferent ($x_{\langle i,j\rangle}$=1), then mention j must be anaphoric (y_j=1).

$$x_{\langle i,j\rangle} \leq y_j \qquad \forall \langle i,j\rangle \in P \quad (8)$$

Resolve anaphors: if a mention is anaphoric (y_j=1), it *must* be coreferent with at least one antecedent.

$$y_j \leq \sum_{i \in M_j} x_{\langle i,j\rangle} \qquad \forall j \in M \quad (9)$$

Do not resolve non-anaphors: if a mention is non-anaphoric (y_j=0), it should have no antecedents.

$$y_j \geq \frac{1}{|M_j|} \sum_{i \in M_j} x_{\langle i,j\rangle} \qquad \forall j \in M \quad (10)$$

These constraints thus directly relate the two tasks. By formulating the problem this way, the decisions of the anaphoricity classifier are not taken on faith as they were with AC-CASCADE. Instead, we optimize over consideration of both possibilities in the objective function (relative to the probability output by the classifier) while ensuring that the final assignments respect the signifance of what it is to be anaphoric or non-anaphoric.

6 Joint Results

Table 2 summarizes the results for these different systems. Both ILP systems are significantly better than the baseline system COREF-PAIRWISE. Despite having lower precision than COREF-PAIRWISE, the COREF-ILP system obtains very large gains in recall to end up with overall f-score gains of 4.3%, 4.2%, and 3.0% across BNEWS, NPAPER, and NWIRE, respectively. The fundamental reason for the increase in recall and drop in precision is that COREF-ILP can posit multiple antecedents for each mention. This is an extra degree of freedom that allows COREF-ILP to cast a wider net, with a consequent risk of capturing incorrect antecedents. Precision is not completely degraded because the optimization performed by ILP utilizes the pairwise probabilities of mention pairs as weights in the objective function to make its assignments. Thus, highly improbable links are still heavily penalized and are not chosen as coreferential.

The JOINT-ILP system demonstrates the benefit ILP's ability to support joint task formulations. It produces significantly better f-scores by regaining some of the ground on precision lost by COREF-ILP. The most likely source of the improved precision of JOINT-ILP is that weights corresponding to the anaphoricity probabilities and constraints (8) and (10) reduce the number of occurrences of non-anaphors being assigned antecedents. There are also improvements in recall over COREF-ILP for NPAPER and NWIRE. A possible source of this difference is constraint (9), which ensures that mentions which are considered anaphoric must have at least one antecedent.

Compared to COREF-PAIRWISE, JOINT-ILP dramatically improves recall with relatively small losses in precision, providing overall f-score gains of 5.3%, 4.9%, and 3.7% on the three datasets.

7 Related Work

As was just demonstrated, ILP provides a principled way to model dependencies between anaphoricity decisions and coreference decisions. In a similar manner, this framework could also be used to capture dependencies among coreference decisions themselves. This option —which we will leave for future work— would make such an approach akin to

System	BNEWS			NPAPER			NWIRE		
	R	P	F	R	P	F	R	P	F
COREF-PAIRWISE	54.4	77.4	63.9	58.1	80.7	67.6	53.8	78.2	63.8
COREF-ILP	62.2	75.5	68.2	67.1	77.3	71.8	60.1	74.8	66.8
JOINT-ILP	62.1	78.0	69.2	68.0	77.6	72.5	60.8	75.8	67.5

Table 2: Recall (R), precision (P), and f-score (F) on the three ACE datasets for the basic coreference system (COREF-PAIRWISE), the coreference only ILP system (COREF-ILP), and the joint anaphoricity-coreference ILP system (JOINT-ILP). All f-score differences are significant ($p < .05$).

a number of recent global approaches.

Luo et al. (2004) use Bell trees to represent the search space of the coreference resolution problem (where each leaf is possible partition). The problem is thus recast as that of finding the "best" path through the tree. Given the rapidly growing size of Bell trees, Luo et al. resort to a beam search algorithm and various pruning strategies, potentially resulting in picking a non-optimal solution. The results provided by Luo et al. are difficult to compare with ours, since they use a different evaluation metric.

Another global approach to coreference is the application of Conditional Random Fields (CRFs) (McCallum and Wellner, 2004). Although both are global approaches, CRFs and ILP have important differences. ILP uses separate local classifiers which are learned without knowledge of the output constraints and are then integrated into a larger inference task. CRFs estimate a global model that directly uses the constraints of the domain. This involves heavy computations which cause CRFs to generally be slow and inefficient (even using dynamic programming). Again, the results presented in McCallum and Wellner (2004) are hard to compare with our own results. They only consider proper names, and they only tackled the task of identifying the correct antecedent only for mentions which have a true antecedent.

A third global approach is offered by Ng (2005), who proposes a global reranking over partitions generated by different coreference systems. This approach proceeds by first generating 54 candidate partitions, which are each generated by a different system. These different coreference systems are obtained as combinations over three different learners (C4.5, Ripper, and Maxent), three sampling methods, two feature sets (Soon et al., 2001; Ng and Cardie, 2002b), and three clustering algorithms (Best-First, Closest-First, and aggressive-merge). The features used by the reranker are of two types: (i) *partition-based* features which are here simple functions of the local features, and (ii) *method-based* features which simply identify the coreference system used for generating the given partition. Although this approach leads to significant gains on the both the MUC and the ACE datasets, it has some weaknesses. Most importantly, the different systems employed for generating the different partitions are all instances of the local classification approach, and they all use very similar features. This renders them likely to make the same types of errors.

The ILP approach could in fact be integrated with these other approaches, potentially realizing the advantages of multiple global systems, with ILP conducting their interactions.

8 Conclusions

We have provided two ILP formulations for resolving coreference and demonstrated their superiority to a pairwise classifier that makes its coreference assignments greedily.

In particular, we have also shown that ILP provides a natural means to express the use of both anaphoricity classification and coreference classification in a single system, and that doing so provides even further performance improvements, specifically f-score improvements of 5.3%, 4.9%, and 3.7% over the base coreference classifier on the ACE datasets.

With ILP, it is not necessary to carefully control the anaphoricity threshold. This is in stark contrast to systems which use the anaphoricity classifier as a filter for the coreference classifier in a cascade setup.

The ILP objective function incorporates the probabilities produced by both classifiers as weights on variables that indicate the ILP assignments for those tasks. The indicator variables associated with those assignments allow several constraints between the tasks to be straightforwardly stated to ensure consistency to the assignments. We thus achieve large improvements with a simple formulation and no fuss. ILP solutions are also obtained very quickly for the objective functions and constraints we use.

In future work, we will explore the use of global constraints, similar to those used by (Barzilay and Lapata, 2006) to improve both precision and recall. For example, we expect transitivity constraints over coreference pairs, as well as constraints on the entire partition (e.g., the number of entities in the document), to help considerably. We will also consider linguistic constraints (e.g., restrictions on pronouns) in order to improve precision.

Acknowledgments

We would like to thank Ray Mooney, Rohit Kate, and the three anonymous reviewers for their comments. This work was supported by NSF grant IIS-0535154.

References

Regina Barzilay and Mirella Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the HLT/NAACL*, pages 359–366, New York, NY.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

A. Kehler, D. Appelt, L. Taylor, and A. Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of HLT/NAACL*, pages 289–296.

Andrew Kehler. 1997. Probabilistic coreference in information extraction. In *Proceedings of EMNLP*, pages 163–173.

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

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.

Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Proceedings of NIPS*.

Joseph F. McCarthy and Wendy G. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of IJCAI*, pages 1050–1055.

Thomas Morton. 1999. Using coreference for question answering. In *Proceedings of ACL Workshop on Coreference and Its Applications*.

Thomas Morton. 2000. Coreference for NLP applications. In *Proceedings of ACL*, Hong Kong.

Vincent Ng and Claire Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of COLING*.

Vincent Ng and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL*, pages 104–111.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve coreference resolution: Issues in representation and optimization. In *Proceedings of ACL*.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of ACL*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*.

Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of ICML*, pages 737–744.

W. Soon, H. Ng, and D. 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 *Proceedings fo the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, CA. Morgan Kaufmann.

Automating Creation of Hierarchical Faceted Metadata Structures

Emilia Stoica and **Marti A. Hearst**
School of Information
UC Berkeley, Berkeley, CA
estoica,hearst@ischool.berkeley.edu

Megan Richardson
Department of Mathematical Sciences
NMSU, Las Cruces, NM
merichar@nmsu.edu

Abstract

We describe Castanet, an algorithm for automatically generating hierarchical faceted metadata from textual descriptions of items, to be incorporated into browsing and navigation interfaces for large information collections. From an existing lexical database (such as WordNet), Castanet carves out a structure that reflects the contents of the target information collection; moderate manual modifications improve the outcome. The algorithm is simple yet effective: a study conducted with 34 information architects finds that Castanet achieves higher quality results than other automated category creation algorithms, and 85% of the study participants said they would like to use the system for their work.

1 Introduction

It is becoming widely accepted that the standard search interface, consisting of a query box and a list of retrieved items, is inadequate for navigation and exploration in large information collections such as online catalogs, digital libraries, and museum image collections. Instead, user interfaces which organize and group retrieval results have been shown to be helpful for and preferred by users over the straight results-list model when engaged in exploratory tasks (Yee et al., 2003; Pratt et al., 1999; Kaki, 2005). In particular, a representation known as hierarchical faceted metadata is gaining great traction within the information architecture and enterprise search communities (Yee et al., 2003; Weinberger, 2005).

A considerable impediment to the wider adoption of collection navigation via metadata in general, and hierarchical faceted metadata in particular, is the need to create the metadata hierarchies and assign the appropriate category labels to the information items. Usually, metadata category structures are manually created by information architects (Rosenfeld and Morville, 2002). While manually created metadata is considered of high quality, it is costly in terms of time and effort to produce, which makes it difficult to scale and keep up with the vast amounts of new content being produced.

In this paper, we describe Castanet, an algorithm that makes considerable progress in automating faceted metadata creation. Castanet creates domain-specific overlays on top of a large general-purpose lexical database, producing surprisingly good results in a matter of minutes for a wide range of subject matter.

In the next section we elaborate on the notion of hierarchical faceted metadata and show how it can be used in interfaces for navigation of information collections. Section 3 describes other algorithms for inducing category structure from textual descriptions. Section 4 describes the Castanet algorithm, Section 5 describes the results of an evaluation with information architects, and Section 6 draws conclusions and discusses future work.

2 Hierarchical Faceted Metadata

A hierarchical faceted metadata system (HFC) creates a set of category hierarchies, each of which corresponds to a different facet (dimension or type). The main application of hierarchical faceted metadata is in user interfaces for browsing and navigating collections of like items.

In the case of a recipe collection, for example, facets may consist of dish type (salad, appetizer), ingredients such as fruits (apricot, apple), vegetables (broccoli, cabbage), meat (beef, fish), preparation method (fry, bake, etc.), calorie count, and so on. Decomposing the description into independent categories allows users to move through large information spaces in a flexible manner. The category metadata guides the user toward possible choices, and organizes the results of keyword searches, allowing users to both refine and expand the current query, while maintaining a consistent representation of the collection's structure. This use of metadata should be integrated with free-text search, allowing the user to follow links, then add search terms, then follow more links, without interrupting the interaction flow.

Usability studies have shown that, when incorporated into a properly-designed user interface, hierarchical faceted metadata provides a flexible, intuitive way to explore a large collection of items that enhances feelings of discovery without inducing a feeling of being lost (Yee et al., 2003).

Note that the HFC representation is intermediate in complexity between that of a monolithic hierarchy and a full-blown ontology. HFC does not capture relations and inferences that are essential for some applications. For example, faceted metadata can express that an image contains a hat and a man and a tree, and perhaps a wearing activity, but does not indicate who is wearing what. This relative simplicity of representation suggests that automatically inferring facet hierarchies may be easier than the full ontology inference problem.

3 Related Work

There is a large literature on document classification and automated text categorization (Sebastiani, 2002). However, that work assumes that the categories of interest are already known, and tries to assign documents to categories. In contrast, in this paper we focus on the problem of determining the categories of interest.

Another thread of work is on finding synonymous terms and word associations, as well as automatic acquisition of IS-A (or genus-head) relations from dictionary definitions and free text (Hearst, 1992; Caraballo, 1999). That work focuses on finding the right position for a word within a lexicon, rather than building up comprehensible and coherent faceted hierarchies.

A major class of solutions for creating subject hierarchies uses data clustering. The Scatter/Gather system (Cutting et al., 1992) uses a greedy global agglomerative clustering algorithm where an initial set of k clusters is recursively re-clustered until only documents remain. Hofmann (1999) proposes the probabilistic latent semantic analysis algorithm (pLSA), a probabilistic version of clustering that uses latent semantic analysis for grouping words and annealed EM for model fitting.

The greatest advantage of clustering is that it is fully automatable and can be easily applied to any text collection. Clustering can also reveal interesting and potentially unexpected or new trends in a group of documents. The disadvantages of clustering include their lack of predictability, their conflation of many dimensions simultaneously, the difficulty of labeling the groups, and the counter-intuitiveness of cluster sub-hierarchies (Pratt et al., 1999).

Blei et al. (2003) developed the LDA (Latent Dirichlet Allocation) method, a generative probabilistic model of discrete data, which creates a hierarchical probabilistic model of documents. It attempts to analyze a text corpus and extract the topics that combined to form its documents. The output of the algorithm was evaluated in terms of perplexity reduction but not in terms of understandability of the topics produced.

Sanderson and Croft (1999) propose a method called *subsumption* for building a hierarchy for a set of documents retrieved for a query. For two terms x and y, x is said to subsume y if the following conditions hold: $P(x|y) \geq 0.8, P(y|x) < 1$. In other words, x subsumes y and is a parent of y, if the documents which contain y, are a subset of the documents which contain x. To evaluate the algorithm the authors asked 8 participants to look at parent-child pairs and state whether or not they were "interesting". Participants found 67% to be interesting as compared to 51% for randomly chosen pairs of words. Of those interesting pairs, 72% were found to display a "type-of" relationship.

Nevill-Manning et.al (1999), Anick et.al (1999) and Vossen (2001) build hierarchies based on substring inclusion. For example, the category *full text indexing and retrieval* is the child of *indexing and retrieval* which in turn is the child of *index*. While these string inclusion approaches expose some structure of the dataset, they can only create subcategories which are substrings of the parent category, which is very restrictive.

Another class of solutions make use of existing lexical hierarchies to build category hierarchies, as we do in this paper. For example, Navigli and Velardi (2003) use WordNet (Fellbaum, 1998) to build a complex ontology consisting of a wide range of relation types (demonstrated on a travel agent domain), as opposed to a set of human-readable hierarchical facets. They develop a complex algorithm for choosing among WordNet senses; it requires building a rich semantic network using WordNet glosses, meronyms, holonyms, and other lexical relations, and using the semantically annotated SemCor collection. The semantic nets are intersected and the correct sense is chosen based on a score assigned to each intersection. Mihalcea and Moldovan (2001) describe a sophisticated method for simplifying WordNet in general, rather than tailoring it to a specific collection.

4 Method

The main idea behind the Castanet algorithm[1] is to carve out a structure from the hypernym (IS-A) relations within the WordNet (Fellbaum, 1998) lexical database. The primary unit of representation in WordNet is the synset, which is a set of words that are considered synonyms for a particular concept. Each synset is linked to other synsets via several types of lexical and semantic relations; we only use hypernymy (IS-A relations) in this algorithm.

[1]A simpler, un-evaluated version of this algorithm was presented previously in a short paper (Stoica and Hearst, 2004).

4.1 Algorithm Overview

The Castanet algorithm assumes that there is text associated with each item in the collection, or at least with a representative subset of the items. The textual descriptions are used *both* to build the facet hierarchies and to assign items (documents, images, citations, etc.) to the facets. The text does not need to be particularly coherent for the algorithm to work; we have applied it to fragmented image annotations and short journal titles, but if the text is impoverished, the information items will not be labeled as thoroughly as desirable and additional manual annotation may be needed.

The algorithm has five major steps:

1. Select target terms from textual descriptions of information items.
2. Build the Core Tree:
   - For each term, if the term is unambiguous (see below), add its synset's IS-A path to the Core Tree.
   - Increment the counts for each node in the synset's path with the number of documents in which the target term appears.
3. Augment the Core Tree with the remaining terms' paths:
   - For each candidate IS-A path for the ambiguous term, choose the path for which there is the most document representation in the Core Tree.
4. Compress the augmented tree.
5. Remove top-level categories, yielding a set of facet hierarchies.

We describe each step in more detail below.

4.2 Select Target Terms

Castanet selects only a subset of terms, called *target terms*, that are intended to best reflect the topics in the documents. Similarly to Sanderson and Croft (1999), we use the *term distribution* – defined as the number of item descriptions containing the term – as the selection criterion. The algorithm retains those terms that have a distribution larger than a threshold and eliminates terms on a stop list. One and two-word consecutive noun phrases are eligible to be considered as terms. Terms that can be adjectives or verbs as well as nouns are optionally deleted.

4.3 Build the Core Tree

The Core Tree acts as the "backbone" for the final category structure. It is built by using paths derived from unambiguous terms, with the goal of biasing the final structure towards the appropriate senses of words.

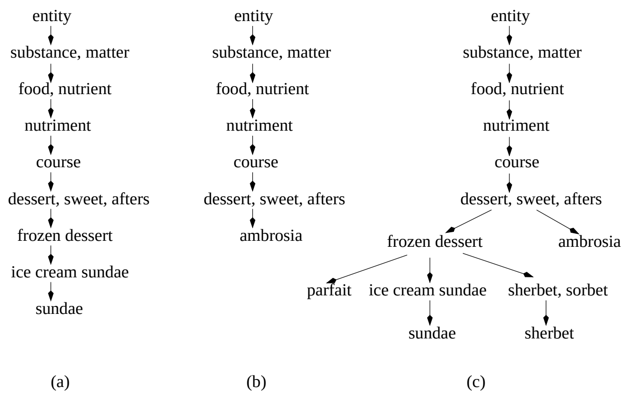


Figure 1: Merging hypernym paths.

4.3.1 Disambiguate using Wordnet Domains

A term is considered unambiguous if it meets at least one of two conditions:

(1) The term has only one sense within WordNet, or
(2) (Optional) The term matches one of the pre-selected WordNet domains (see below).

From our experiments, about half of the eligible terms have only one sense within WordNet. For the rest of terms, we disambiguate between multiple senses as follows.

WordNet provides a cross-categorization mechanism known as *domains*, whereby some synsets are assigned general category labels. However, only a small subset of the nouns in WordNet have domains assigned to them. For example, for a medicine collection, we found that only 4% of the terms have domains *medicine* or *biology* associated with them. For this reason, we use an additional resource called *Wordnet Domains* (Magnini, 2000), which assigns domains to WordNet synsets. In this resource, *every* noun synset in WordNet has been semi-automatically annotated with one of about 200 Dewey Decimal Classification labels. Examples include *history, literature, plastic arts, zoology,* etc.

In Castanet, Wordnet Domains are used as follows. First, the system counts how many times each domain is represented by target terms, building a list of the most well-represented domains for the collection. Then, in a manual intervention step, the information architect selects the subset of the well-represented domains which are meaningful for the collection in question.

For example, for a collection of biomedical journal titles, *Surgery* should be selected as a domain, whereas for an art history image collection, *Architecture* might be chosen. When processing the word *lancet*, the choice of domain distinguishes between the hyponym path *entity* $\rightarrow$ *object* $\rightarrow$ *artifact* $\rightarrow$ *instrumentality* $\rightarrow$ *device* $\rightarrow$ *instrument* $\rightarrow$ *medical instrument* $\rightarrow$ *surgical instrument*

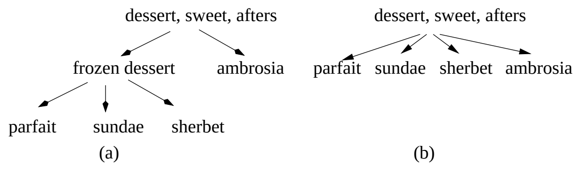


Figure 2: Compressing the tree.

$\rightarrow$ *lancet* and *entity* $\rightarrow$ *object* $\rightarrow$ *artifact* $\rightarrow$ *structure, construction* $\rightarrow$ *arch* $\rightarrow$ *pointed arch* $\rightarrow$ *Gothic arch* $\rightarrow$ *lancet arch, lancet* $\rightarrow$ *lancet*.

In some cases, more than one domain may be relevant for a given term and for a given collection. For example, the term *brain* is annotated with two domains, *Anatomy* and *Psychology*, which are both relevant domains for a biomedical journal collection. Currently for these cases the algorithm breaks the tie by choosing the sense with the lowest WordNet sense number (corresponding to the most common sense), which in this case selects the *Anatomy* sense. However, we see this forced choice as a limitation, and in future work we plan to explore how to allow a term to have more than one occurrence in the metadata hierarchies.

4.3.2 Add Paths to Core Tree

To build the Core Tree, the algorithm marches down the list of unambiguous terms and for each term looks up its synset and its hypernym path in WordNet. (If a term does not have representation in WordNet, then it is not included in the category structure.) To add a path to the Core Tree, its path is merged with those paths that have already been placed in the tree. Figure 1(a-b) shows the hypernym paths for the synsets corresponding to the terms *sundae* and *ambrosia*. Note that they have several hypernym path nodes in common: *(entity), (substance, matter), (food, nutrient), (nutriment), (course), (dessert, sweet, afters)*. Those shared paths are merged by the algorithm; the results, along with the paths for *parfait* and *sherbert* are shown in Figure 1(c).

In addition to augmenting the nodes in the tree, adding in a new term increases a count associated with each node on its path; this count corresponds to how many documents the term occurs in. Thus the more common a term, the more weight it places on the path it falls within.

4.4 Augment the Core Tree / Disambiguate Terms

The Core Tree contains only a subset of terms in the collection (those that have only one path or whose sense can be selected with WordNet Domains). The next step is to add in the paths for the remaining target terms which are ambiguous according to WordNet.

The Core Tree is built with a bias towards paths that are most likely to be appropriate for the collection as a whole. When confronted with a term that has multiple possible IS-A paths corresponding to multiple senses, the system favors the more common path over other alternatives.

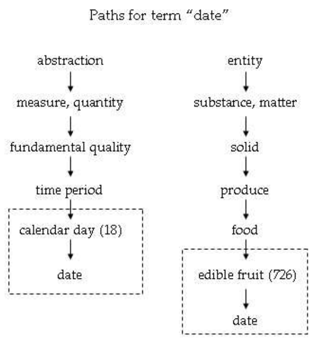


Figure 3: Two path choices for an ambiguous term.

Assume that we want to add the term *date* to the Core Tree for a collection of recipes, and that currently there are two paths corresponding to two of its senses in the Core Tree (see Figure 3). To decide which of the two paths to merge *date* into, the algorithm looks at the number of items assigned to the deepest node that is held in common between the existing Core Tree and each candidate path for the ambiguous term. The path for the *calendar day* sense has fewer than 20 documents assigned to it (corresponding to terms like *Valentine's Day*), whereas the path for the *edible fruit* sense has more than 700 documents assigned. Thus *date* is added to the fruit sense path. (The counts for the ambiguous terms' document hits are *not* incorporated into the new tree.)

Also, to eliminate unlikely senses, each candidate sense's hypernym path is required to share at least $j\%$ of its nodes with nodes already in the Core Tree, where the user sets j (usually between 40 and 60%). Thus the romantic appointment sense of *date* would not be considered as most of its hypernym path is not in the Core Tree. If no path passes the threshold, then the first sense's hypernym path (according to WordNet's sense ordering) is placed in the tree.

4.5 Compress the Tree

The tree that is obtained in the previous step usually is very deep, which is undesirable from a user interface perspective. Castanet uses two rules for compressing the tree:

1. Starting from the leaves, recursively eliminate a parent that has fewer than k children, unless the parent is the root or has an item count larger than $0.1\times$(maximum term distribution).
2. Eliminate a child whose name appears within the parent's name, unless the child contains a WordNet domain name.

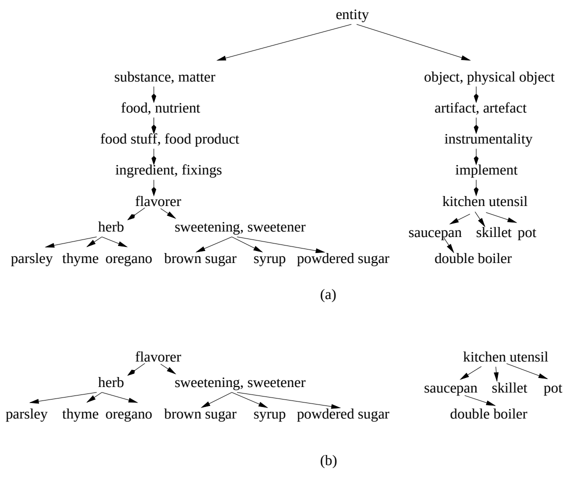


Figure 4: Eliminating top levels.

For example, consider the tree in Figure 1(c) and assume that $k = 2$, which means eliminate parents that have fewer than two children.

Starting from the leaves, by applying Rule 2, nodes (*ice cream sundae*), (*sherbet, sorbet*), (*course*), (*nutriment*), (*food, nutrient*), (*substance, matter*) and (*entity*) are eliminated since they have only one child. Figure 2(a) shows the resulting tree. Next, by applying Rule 3, the node *frozen dessert* is eliminated, since it contains the word *dessert* which also appears in the name of its parent. The final tree is presented in Figure 2(b). Note that this is a rather aggressive compression strategy, and the algorithm can be adjusted to allow more hierarchy to be retained.

4.6 Prune Top Level Categories / Create Facets

The final step is to create a set of facet sub-hierarchies. The goal is to create a moderate set of facets, each of which has moderate depth and breadth at each level, in order to enhance the navigability of the categories. Pruning the top levels can be automated, but a manual editing pass over the outcome will produce the best results.

To eliminate the top levels in an automated fashion, for each of the nine tree roots in the WordNet noun database, manually cut the top t levels (where $t = 4$ for the recipes collection). Then, for each of the resulting trees, recursively test if its root has more than $n = 6$ children. If it does, then the tree is considered a facet; otherwise, the current root is deleted and the algorithm tests to see if each new root has n children. Those subtrees that do not meet the criterion are omitted from the final set of facets.

Consider the tree in Figure 4(a). In this case, the categories of interest are (*flavorer*) and (*kitchen utensil*) along with their children. However, to reach any of these categories, the user has to descend six levels, each of which has very little information. Figure 4(b) shows the resulting facets, which (subjectively) are at an informative level of description for an information architecture. (In this illustration, $t = 2$.)

Often the internal nodes of WordNet paths do not have the most felicitous names, e.g., *edible fruit* instead of *fruit*. Although we did not edit these names for the usability study, it is advisable to do so.

5 Evaluation

The intended users of the Castanet algorithm are information architects and others who need to build structures for information collections. A successful algorithm must be perceived by information architects as making their job easier. If the proposed category system appears to require a lot of work to modify, then IAs are likely to reject it. Thus, to evaluate Castanet's output, we recruited information architects and asked them to compare it to one other state-of-the-art approach as well as a baseline. The participants were asked to assess the qualities of each category system and to express how likely they would be to use each in their work.

5.1 Study Design

The study compared the output of four algorithms: (a) Baseline (frequent words and two-word phrases), (b) Castanet, (c) LDA (Blei et al., 2003)[2] and (d) Subsumption (Sanderson and Croft, 1999). The algorithms were applied to a dataset of $13,000$ recipes from Southwestcooking.com. Participants were recruited via email and were required to have experience building information architectures and to be at least familiar with recipe websites (to show their interest in the domain).

Currently there are no standard tools used by information architects for building category systems from free text. Based on our own experience, we assumed a strong baseline would be a list of the most frequent words and two-word phrases (stopwords removed); the study results confirmed this assumption. The challenge for an automated system is to be preferred to the baseline.

The study design was within-participants, where each participant evaluated Castanet, a Baseline approach, and either Subsumption (N=16) or LDA (N=18).[3] Order of showing Castanet and the alternative algorithm was counterbalanced across participants in each condition.

Because the algorithms produce a large number of hierarchical categories, the output was shown to the

[2]Using code by Blei from www.cs.princeton.edu/˜blei/lda-c/

[3]Pilot studies found that participants became very frustrated when asked to compare LDA against Subsumption, since neither tested well, so we dropped this condition. We did not consider asking any participant to evaluate all three systems, to avoid fatigue. To avoid biasing participants towards any approach, the target algorithms were given the neutral names of Pine, Birch, and Oak. Castanet was run without Domains for a fairer comparison. Top level pruning was done automatically as described, but with a few manual adjustments.

	Cas.	Bas.	LDA	Cas.	Bas.	Sub.
Def. Yes	4	2	0	2	2	0
Yes	10	10	0	13	11	6
No	2	2	2	1	3	2
Def. No	2	4	16	0	0	8

Table 1: Responses to the question "Would you be likely to use this algorithm in your work?" comparing Castanet to the Baseline and LDA (N=18), and comparing Castanet to the Baseline and Subsumption (N=16).

	Cas. (34)	LDA (18)	Sub. (16)
Meaningful	2.9	1.2	1.8
Systematic	2.8	1.4	1.8
Import. Concepts	2.8	1.3	1.9

Table 2: Average responses to questions about the quality of the category systems. N shown in parentheses. Assessed on a four point scale where higher is better.

participants using the open source Flamenco collection browser[4] (see Figure 5). Clicking on a link shows subcategories as well as items that have been assigned that category. For example, clicking on the *Penne* subcategory beneath *Pasta* in the Castanet condition shows 5 recipes that contain the word *penne* as well as the other categories that have been assigned to these recipes. Since LDA does not create names for its output groups, they were assigned the generic names Category 1, 2, etc. Assignment of categories to items was done on a strict word-match basis; participants were not asked to assess the item assignment aspect of the interface.

At the start of the study, participants answered questions about their experience designing information architectures. They were then asked to look at a partial list of recipes and think briefly about what their goals would be in building a website for navigating the collection.

Next they viewed an ordered list of frequent terms drawn automatically from the collection (Baseline condition). After this, they viewed the output of one of the two target category systems. For each algorithm, participants were asked questions about the top-level categories, such as *Would you add any categories?* (possible responses: (a) No, None, (b) Yes, one or two, (c) Yes, a few, and (d) Yes, many). They were then asked to examine two specific top level categories in depth (e.g., *For category Bread, would you remove any subcategories?*). At the end of each assessment, they were asked to comment on general aspects of the category system as a whole (discussed below). After having seen both category systems, participants were asked to state how likely they would be to use the algorithm (e.g., *Would you use Oak? Would you use Birch? Would you use the frequent words list?*) Answer types were (a) No, definitely not, (b) Probably not, (c) Yes, I might want to use this system in some cases, and (d) Yes, I would definitely use this system.

[4]Available at flamenco.berkeley.edu

5.2 Results

Table 1 shows the responses to the final question about how likely the participants are to use the results of each algorithm for their work. Both Castanet and the Baseline fare well, with Castanet doing somewhat better. 85% of the Castanet evaluators said yes or definitely yes to using it, compared to 74% for the Baseline. Only one participant said "no" to Castanet but "yes" to the Baseline, suggesting that both kinds of information are useful for information architects.

The comparison algorithms did poorly. Subsumption received 38% answering "yes" or "definitely yes" to the question about likelihood of use. LDA was rejected by all participants. A t-test (after converting responses to a 1-4 scale) shows that Castanet obtains significantly better scores than LDA (t = 7.88 > 2.75) and Subsumption (t = 4.50 > 2.75), for p = 0.005. The differences between Castanet and the Baseline are not significant.

Table 2 shows the average responses to the questions *(i) Overall, these are categories meaningful; (ii) Overall, these categories describe the collection in a systematic way; (iii) These categories capture the important concepts.*) They were scored as 1= Strongly disagree, 2 = Disagree Somewhat, 3 = Agree Somewhat, and 4 = Strongly agree. Castanet's score was about 35% higher than Subsumption's, and about 50% higher than LDA's.

Participants were asked to scrutinize the top-level categories and assess whether they would add categories, remove some, merge or rename some. The ratings were again converted to a four point scale (no changes = 4, change one or two = 3, change a few = 2, change many = 1). Table 3 shows the results. Castanet scores as well as or better than the others on all measures except Rename; Subsumption scores slightly higher on this measure, and does well on Split as well, but very poorly on Remove, reflecting the fact that it produces well-named categories at the top level, but too many at too fine a granularity.

Participants were also asked to examine two subcategories in detail. Table 4 shows results averaged across the two subcategories for number of categories to add, remove, promote, move, and how well the subcategories matched their expectations. Castanet performs especially well on this last measure (2.5 versus 1.5 and 1.7). Participants generally did not suggest moves or promotions.

Thus on all measures, we see Castanet outperforming the other state-of-the-art algorithms. Note that we did not explicitly evaluate the "facetedness" of the category systems, as we thought this would be too difficult for the participants to do. We feel the questions about the coher-

	Cas. (34).	LDA (18)	Sub. (16)
Add	2.8	2.6	2.0
Remove	2.3	2.4	1.9
Rename	2.7	2.7	3.3
Merge	2.7	2.5	2.4
Split	3.8	3.3	3.8

Table 3: Assessing top-level categories.

	Cas. (34).	LDA (18)	Sub. (16)
Add	2.8	2.8	2.4
Remove	3.4	2.2	2.5
Promote	3.7	3.4	3.8
Move	3.8	3.3	3.6
Matched Exp.	2.5	1.5	1.7

Table 4: Assessing second-level categories.

ence, systematicity, and coverage of the category systems captured this to some degree.

6 Conclusions and Future Work

We have presented an algorithm called Castanet that creates hierarchical faceted metadata using WordNet and Wordnet Domains. A questionnaire revealed that 85% information architects thought it was likely to be useful, compared to 0% for LDA and 38% for Subsumption. Although not discussed here, we have successfully applied the algorithm to other domains including biomedical journal titles and art history image descriptions, and to another lexical hierarchy, MeSH.[5]

Although quite useful "out of the box," the algorithm could benefit by several improvements and additions. The processing of the terms should recognize spelling variations (such as aging vs. ageing) and morphological variations. Verbs and adjectives are often quite important for a collection (e.g., stir-fry for cooking) and should be included, but with caution. Some terms should be allowed to occur with more than one sense if this is required by the dataset (and some in more than one facet even with the same sense, as seen in the *brain* example). Currently if a term is in a document it is assumed to use the sense assigned in the facet hierarchies; this is often incorrect, and so terms should be disambiguated within the text before automatic category assignment is done. And finally, WordNet is not exhaustive and some mechanism is needed to improve coverage for unknown terms.

Acknowledgements Thanks to Lou Rosenfeld and Rashmi Sinha for their help finding participants, and to all the participants themselves. This work was funded in part by NSF DBI-0317510 and in part by the Summer Undergraduate Program in Engineering Research at Berkeley (SUPERB).

[5]MEdical Subject Headings, http://www.nlm.nih.gov/mesh/

References

Peter Anick and Susesh Tipirneni. 1999. The paraphrase search assistant:terminological feedback for iterative information seeking. In *Procs. of SIGIR'99*.

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

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *ACL '99*.

Douglas Cutting, David Karger D., Jan Pedersen, and John W. Tukey. 1992. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proc. of SIGIR'92*.

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

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING '92*.

Thomas Hofmann. 1999. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *Procs. of IJCAI'99*, Stolckholm, July.

Mika Kaki. 2005. Findex: Search result categories help users when document ranking fails. In *Proc. of CHI '05*.

Bernardo Magnini. 2000. Integrating subject field codes into WordNet. In *Procs. of LREC 2000*, Athens, Greece.

Rada Mihalcea and Dan I. Moldovan. 2001. Ez.wordnet: Principles for automatic generation of a coarse grained wordnet. In *Procs. of FLAIRS Conference 2001*, May.

Roberto Navigli, Paola Velardi, and Aldo Gangemi. 2003. Ontology learning and its application to automated terminology translation. *Intelligent Systems*, 18(1):22–31.

Craig Nevill-Manning, I. Witten, and G. Paynter. 1999. Lexically generated subject hierarchies for browsing large collections. *Inter. J. on Digital Libraries*, 2(2+3):111–123.

Wanda Pratt, Marti Hearst, and Larry Fagan. 1999. A knowledge-based approach to organizing retrieved documents. In *Procs. of AAAI 99*, Orlando, FL.

Louis Rosenfeld and Peter Morville. 2002. *Information Architecture for the World Wide Web: Designing Large-scale Web Sites*. O'Reilly & Associates, Inc.

Mark Sanderson and Bruce Croft. 1999. Deriving concept hierarchies from text. In *Procs. of SIGIR '99*.

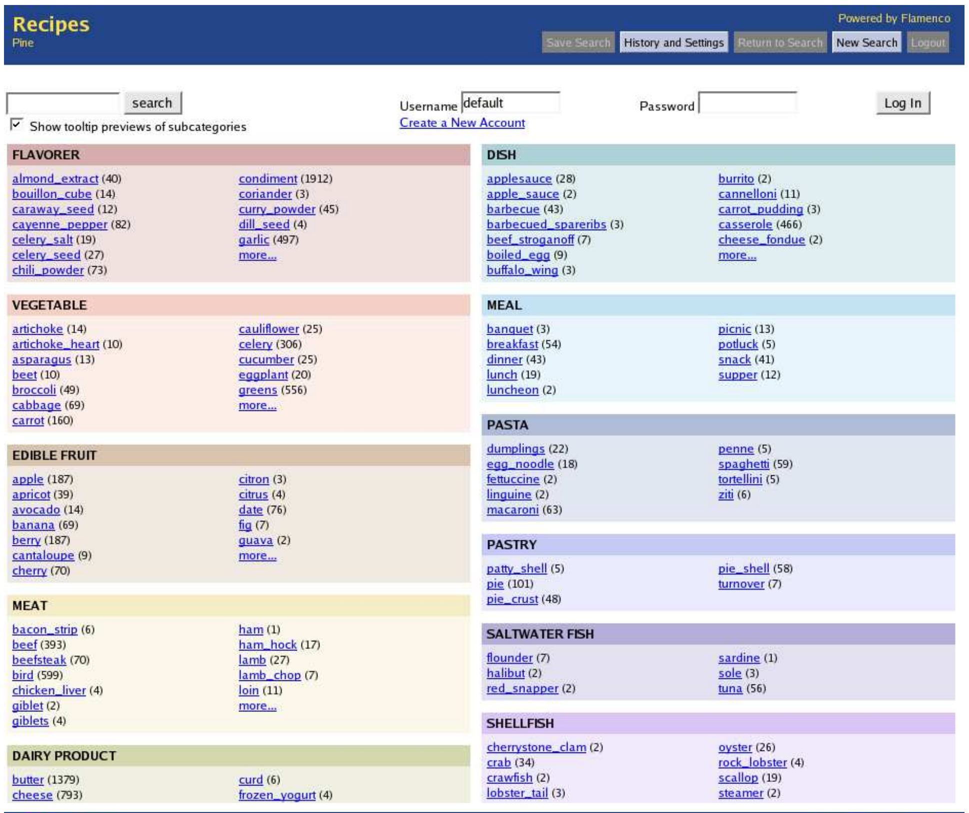
Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Emilia Stoica and Marti Hearst. 2004. Nearly-automated metadata hierarchy creation. In *Proc. of HLT-NAACL 2004*.

Piek Vossen. 2001. Extending, trimming and fussing wordnet for technical documents. In *NAACL 2001 Workshop and Other Lexical Resources*, East Stroudsburg, PA.

Dave Weinberger. 2005. Taxonomies and tags: From trees to piles of leaves. In *Release 1.0*, Feb.

Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. 2003. Faceted metadata for image search and browsing. In *Procs. of CHI '03*, Fort Lauderdale, FL, April.



(a)

Recipes
Pine

Username default
Password
Create a New Account
✓ Show tooltip previews of subcategories

FLAVORER
almond_extract (40), bouillon_cube (14), caraway_seed (12), cayenne_pepper (82), celery_salt (19), celery_seed (27), chili_powder (73), condiment (1912), coriander (3), curry_powder (45), dill_seed (4), garlic (497), more...

VEGETABLE
artichoke (14), artichoke_heart (10), asparagus (13), beet (10), broccoli (49), cabbage (69), carrot (160), cauliflower (25), celery (306), cucumber (25), eggplant (20), greens (556), more...

EDIBLE FRUIT
apple (187), apricot (39), avocado (14), banana (69), berry (187), cantaloupe (9), cherry (70), citron (3), citrus (4), date (76), fig (7), guava (2), more...

MEAT
bacon_strip (6), beef (393), beefsteak (70), bird (599), chicken_liver (4), giblet (2), giblets (4), ham (1), ham_hock (17), lamb (27), lamb_chop (7), loin (11), more...

DAIRY PRODUCT
butter (1379), cheese (793), curd (6), frozen_yogurt (4)

DISH
applesauce (28), apple_sauce (2), barbecue (43), barbecued_spareribs (3), beef_stroganoff (7), boiled_egg (9), buffalo_wing (3), burrito (2), cannelloni (11), carrot_pudding (3), casserole (466), cheese_fondue (2), more...

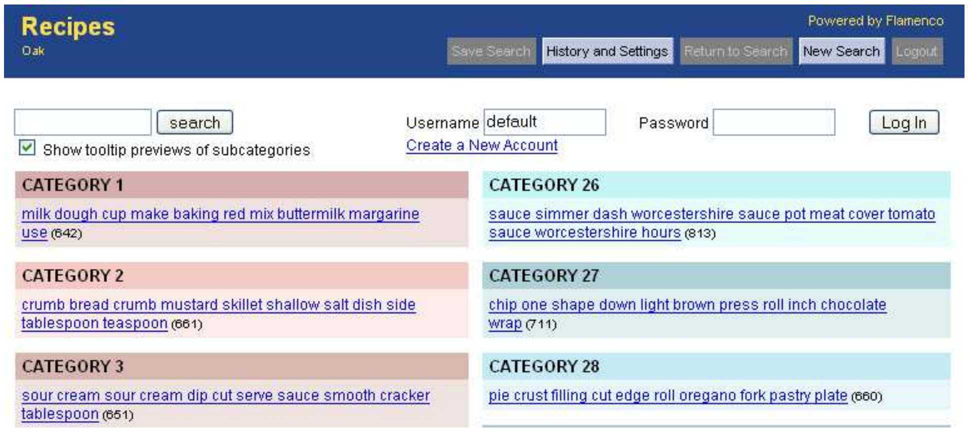
MEAL
banquet (3), breakfast (54), dinner (43), lunch (19), luncheon (2), picnic (13), potluck (5), snack (41), supper (12)

PASTA
dumplings (22), egg_noodle (18), fettuccine (2), linguine (2), macaroni (63), penne (5), spaghetti (59), tortellini (5), ziti (6)

PASTRY
patty_shell (5), pie (101), pie_crust (48), pie_shell (58), turnover (7)

SALTWATER FISH
flounder (7), halibut (2), red_snapper (2), sardine (1), sole (3), tuna (56)

SHELLFISH
cherrystone_clam (2), crab (34), crawfish (2), lobster_tail (3), oyster (26), rock_lobster (4), scallop (19), steamer (2)



(b)

Recipes
Oak

Username default
Password
Create a New Account
✓ Show tooltip previews of subcategories

CATEGORY 1
milk dough cup make baking red mix buttermilk margarine use (642)

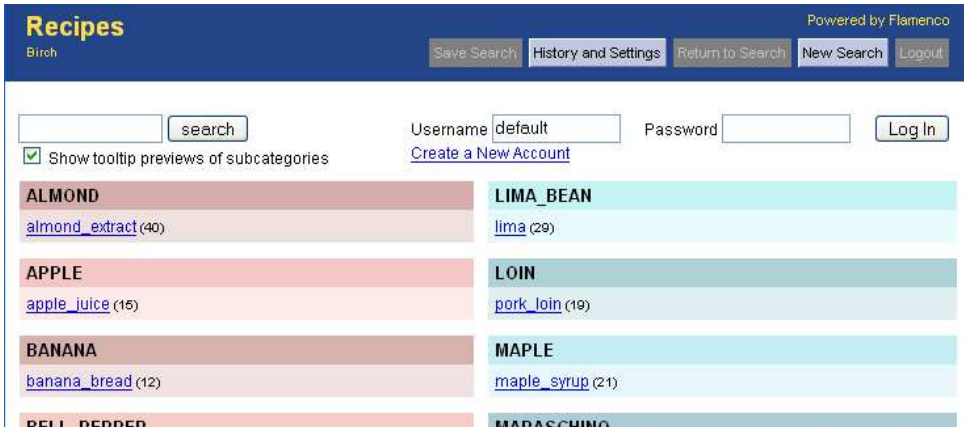
CATEGORY 2
crumb bread crumb mustard skillet shallow salt dish side tablespoon teaspoon (661)

CATEGORY 3
sour cream sour cream dip cut serve sauce smooth cracker tablespoon (651)

CATEGORY 26
sauce simmer dash worcestershire sauce pot meat cover tomato sauce worcestershire hours (813)

CATEGORY 27
chip one shape down light brown press roll inch chocolate wrap (711)

CATEGORY 28
pie crust filling cut edge roll oregano fork pastry plate (660)



(c)

Recipes
Birch

Username default
Password
Create a New Account
✓ Show tooltip previews of subcategories

ALMOND
almond_extract (40)

APPLE
apple_juice (15)

BANANA
banana_bread (12)

BELL PEPPER

LIMA_BEAN
lima (29)

LOIN
pork_loin (19)

MAPLE
maple_syrup (21)

MARASCHINO

Figure 5: Partial view of categories obtained by (a) Castanet, (b) LDA and (c) Subsumption on the Recipes collection, displayed in the Flamenco interface.

Cross-Instance Tuning of Unsupervised Document Clustering Algorithms[*]

Damianos Karakos, Jason Eisner
and Sanjeev Khudanpur
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
{damianos,eisner,khudanpur}@jhu.edu

Carey E. Priebe
Dept. of Applied Mathematics
and Statistics
Johns Hopkins University
Baltimore, MD 21218
cep@jhu.edu

Abstract

In unsupervised learning, where no training takes place, one simply hopes that the unsupervised learner will work well on *any* unlabeled test collection. However, when the variability in the data is large, such hope may be unrealistic; a *tuning* of the unsupervised algorithm may then be necessary in order to perform well on new test collections. In this paper, we show how to perform such a tuning in the context of unsupervised document clustering, by (i) introducing a degree of freedom, α, into two leading information-theoretic clustering algorithms, through the use of generalized mutual information quantities; and (ii) selecting the value of α based on clusterings of similar, but *supervised* document collections (cross-instance tuning). One option is to perform a tuning that directly minimizes the error on the supervised data sets; another option is to use "strapping" (Eisner and Karakos, 2005), which builds a classifier that learns to distinguish good from bad clusterings, and then selects the α with the best predicted clustering on the test set. Experiments from the "20 Newsgroups" corpus show that, although both techniques improve the performance of the baseline algorithms, "strapping" is clearly a better choice for cross-instance tuning.

[*]This work was partially supported by the DARPA GALE program (Contract № HR0011-06-2-0001) and by the JHU WSE/APL Partnership Fund.

1 Introduction

The problem of combining labeled and unlabeled examples in a learning task *(semi-supervised learning)* has been studied in the literature under various guises. A variety of algorithms (e.g., bootstrapping (Yarowsky, 1995), co-training (Blum and Mitchell, 1998), alternating structure optimization (Ando and Zhang, 2005), etc.) have been developed in order to improve the performance of supervised algorithms, by automatically extracting knowledge from lots of *unlabeled* examples. Of special interest is the work of Ando and Zhang (2005), where the goal is to build many supervised auxiliary tasks from the unsupervised data, by creating artificial labels; this procedure helps learn a transformation of the input space that captures the relatedness of the auxiliary problems to the task at hand. In essence, Ando and Zhang (2005) transform the semi-supervised learning problem to a *multi-task learning* problem; in multi-task learning, a (usually large) set of *supervised* tasks is available for training, and the goal is to build models which can *simultaneously* do well on all of them (Caruana, 1997; Ben-David and Schuller, 2003; Evgeniou and Pontil, 2004).

Little work, however, has been devoted to study the situation where lots of labeled examples, of one kind, are used to build a model which is tested on unlabeled data of a "different" kind. This problem, which is the topic of this paper, cannot be cast as a multi-task learning problem (since there are labeled examples of only one kind), neither can be cast as a semi-supervised problem (since there are no training labels for the test task). Note that we are interested in the case where the hidden test labels may have no semantic relationship with the training labels; in some cases, there may not even be any information about the test labels—what they represent, how many they are, or at what granularity they describe the data. This situation can arise in the case of unsupervised clustering of documents from a large and diverse corpus: it may not be known in what way the resulting clusters split the corpus (is it in terms of topic? genre? style? authorship? a combination of the above?), unless one inspects each resulting cluster to determine its "meaning."

At this point, we would like to differentiate between two concepts: a target *task* refers to a class of problems that have a common, high-level description (e.g., the text document clustering task, the speech recognition task, etc.). On the other hand, a task *instance* refers to a particular example from the class. For instance, if the task is *"document clustering,"* a task instance could be *"clustering of a set of scientific documents into particular fields"*; or, if the task is *"parsing,"* a task instance could be *"parsing of English sentences from the Wall Street Journal corpus"*. For the purposes of this paper, we further assume that there are task instances which are *unrelated*, in the sense that that there are no common labels between them. For example, if the task is *"clustering from the 20 Newsgroups corpus,"* then *"clustering of the computer-related documents into PC-related and Mac-related"* and *"clustering of the politics-related documents into Middle-East-related and non-Middle-East-related"* are two distinct, unrelated instances. In more mathematical terms, if task instances T_1, T_2 take sets of observations $\boldsymbol{X}_1, \boldsymbol{X}_2$ as input, and try to predict labels from sets S_1, S_2, respectively, then they are called unrelated if $X_1 \cap X_2 = \varnothing$ and $S_1 \cap S_2 = \varnothing$.

The focus of this paper is to study the problem of *cross-instance tuning* of unsupervised algorithms: how one can tune an algorithm, which is used to solve a particular task instance, using knowledge from an unrelated task instance. To the best of our knowledge, this cross-instance learning problem has only been tackled in (Eisner and Karakos, 2005), whose "strapping" procedure learns a meta-classifier for distinguishing good from bad clusterings.

In this paper, we introduce a scalar parameter α (a new degree of freedom) into two basic unsupervised clustering algorithms. We can tune α to maximize unsupervised clustering performance on *different* task instances where the correct clustering is known. The hope is that tuning the parameter learns something about the task in general, which transfers from the supervised task instances to the unsupervised one. Alternatively, we can tune a meta-classifier so as to select good values of α on the supervised task instances, and then use the same meta-classifier to select a good (possibly different) value of α in the unsupervised case.

The paper is organized as follows: Section 2 gives a background on text categorization, and briefly describes the algorithms that we use in our experiments. Section 3 describes our parameterization of the clustering algorithms using Jensen-Rényi divergence and Csiszár's mutual information. Experimental results from the "20 Newsgroups" data set are shown in Section 4, along with two techniques for cross-instance learning: (i) "strapping," which, at test time, picks a parameter based on various "goodness" cues that were learned from the labeled data set, and (ii) learning the parameter from a supervised data set which is chosen to statistically match the test set. Finally, concluding remarks appear in Section 5.

2 Document Categorization

Document categorization is the task of deciding whether a piece of text belongs to any of a set of prespecified categories. It is a generic text processing task useful in indexing documents for later retrieval, as a stage in natural language processing systems, for content analysis, and in many other roles (Lewis and Hayes, 1994). Here, we deal with the unsupervised version of document categorization, in which we are interested in clustering together documents which (hopefully) belong to the same topic, without having any training examples.[1] *Supervised* information-theoretic clustering approaches (Torkkola, 2002; Dhillon et al., 2003) have been shown to be very effective, even with a small amount of labeled data, while *unsupervised* methods (which are of particular interest to us) have been shown to be competitive, matching the classification accuracy of supervised methods.

Our focus in this paper is on document categorization algorithms which use information-theoretic

[1]By this, we mean that training examples having the same category labels as the test examples are not available.

criteria, since there are natural ways of generalizing these criteria through the introduction of tunable parameters. We use two such algorithms in our experiments, the sequential Information Bottleneck (sIB) and Iterative Denoising Trees (IDTs); details about these algorithms appear below.

A note on mathematical notation: We assume that we have a collection $\mathcal{A} = \{X(1), \ldots, X(N)\}$ of N documents. Each document $X(i)$ is essentially a "bag of words", and induces an empirical distribution $\hat{P}_{X(i)}$ on the vocabulary $\mathcal{X}$. Given a subset (cluster) C of documents, the conditional distribution on $\mathcal{X}$, given the cluster, is just the centroid: $\hat{P}_{X|C} = \frac{1}{|C|} \sum_{X(i) \in C} \hat{P}_{X(i)}$. If a subcollection $S \subset \mathcal{A}$ of documents is partitioned into clusters $C_1, \ldots, C_m$, and each document $X(i) \in S$ is assigned to a cluster $C_{Z(i)}$, where $Z(i) \in \{1, \ldots, m\}$ is the cluster index, then the mutual information between words and corresponding clusters is given by

$$I(X; Z|S) = \sum_{z \in \{1,\ldots,m\}} P(z|S) D(\hat{P}_{X|C_z} \| \hat{P}_{X|S}),$$

where $P(z|S) \triangleq |C_z|/|S|$ is the "prior" distribution on the clusters and $D(\cdot\|\cdot)$ is the Kullback-Leibler divergence (Cover and Thomas, 1991).

2.1 The Information Bottleneck Method

The Information Bottleneck (IB) method (Tishby et al., 1999; Slonim and Tishby, 2000; Slonim et al., 2002) is one popular approach to unsupervised categorization. The goal of the IB (with "hard" clustering) is to find clusters such that the mutual information $I(X; Z)$ between words and clusters is as large as possible, under a constraint on the number of clusters. The procedure for finding the maximizing clustering in (Slonim and Tishby, 2000) is agglomerative clustering, while in (Slonim et al., 2002) it is based on many random clusterings, combined with a sequential update algorithm, similar to K-means. The update algorithm re-assigns each data point (document) d to its most "similar" cluster C, in order to *minimize* $I(X; Z|C \cup \{d\})$, i.e.,

$$\delta D(\hat{P}_{X|\{d\}} \| \hat{P}_{X|\{d\} \cup C}) + (1-\delta) D(\hat{P}_{X|C} \| \hat{P}_{X|\{d\} \cup C}),$$

where $\delta = \frac{1}{|C|+1}$. This latter procedure is called the *sequential Information Bottleneck* (sIB) method, and is considered the state-of-the-art in unsupervised document categorization.

2.2 Iterative Denoising Trees

Decision trees are a powerful technique for equivalence classification, accomplished through a recursive successive refinement (Jelinek, 1997). In the context of unsupervised classification, the goal of decision trees is to cluster empirical distributions (bags of words) into a given number of classes, with each class corresponding to a leaf in the tree. They are built top-down (as opposed to the bottom-up construction in IB) using maximization of mutual information between words and clusters $I(X; Z|t)$ to drive the splitting of each node t; the hope is that each leaf will contain data points which belong to only one latent category.

Iterative Denoising Trees (also called Integrated Sensing and Processing Decision Trees) were introduced in (Priebe et al., 2004a), as an extension of regular decision trees. Their main feature is that they *transform* the data at each node, before splitting, by projecting into a low-dimensional space. This transformation corresponds to feature extraction; different features are suppressed (or amplified) by each transformation, depending on what data points fall into each node (*corpus-dependent-feature-extraction* property (Priebe et al., 2004b)). Thus, dimensionality reduction and clustering are chosen so that they *jointly* optimize the local objective.

In (Karakos et al., 2005), IDTs were used for an unsupervised hyperspectral image segmentation application. The objective at each node t was to maximize the mutual information between spectral components and clusters given the pixels at node t, computed from the *projected* empirical distributions. At each step of the tree-growing procedure, the node which yielded the highest increase in the average, per-node mutual information, was selected for splitting (until a desired number of leaves was reached). In (Karakos et al., 2007b), the mutual information objective was replaced with a parameterized form of mutual information, namely the Jensen-Rényi divergence (Hero et al., 2001; Hamza and Krim, 2003), of which more details are provided in the next section.

3 Parameterizing Unsupervised Clustering

As mentioned above, the algorithms considered in this paper (sIB and IDTs) are unsupervised, in the sense that they can be applied to test data without any need for tuning. Our procedure of adapting them, based on some supervision on a different task instance, is by introducing a parameter into the unsupervised algorithm. At least for simple cross-instance tuning, this parameter represents the information which is passed between the supervised and the unsupervised instances.

The parameterizations that we focused on have to do with the information-theoretic *objectives* in the two unsupervised algorithms. Specifically, following (Karakos et al., 2007b), we replace the mutual information quantities in IDTs as well as sIB with the *parameterized* mutual information measures mentioned above. These two quantities provide estimates of the dependence between the random quantities in their arguments, just as the usual mutual information does, but also have a scalar parameter $\alpha \in (0, 1]$ that controls the sensitivity of the computed dependence on the details of the joint distribution of X and Z. As a result, the effect of data sparseness on estimation of the joint distribution can be mitigated when computing these measures.

3.1 Jensen-Rényi Divergence

The Jensen-Rényi divergence was used in (Hero et al., 2001; Hamza and Krim, 2003) as a measure of similarity for image classification and retrieval. For two discrete random variables X, Z with distributions P_X, P_Z and conditional $P_{X|Z}$, it is defined as

$$I_\alpha(X; Z) = H_\alpha(P_X) - \sum_z P_Z(z) H_\alpha(P_{X|Z}(\cdot|z)), \tag{1}$$

where $H_\alpha(\cdot)$ is the Rényi entropy, given by

$$H_\alpha(P) = \frac{1}{1-\alpha} \log \left(\sum_{x \in \mathcal{X}} P(x)^\alpha \right), \quad \alpha \neq 1. \tag{2}$$

If $\alpha \in (0, 1)$, H_α is a concave function, and hence $I_\alpha(X; Z)$ is non-negative (and it is equal to zero if and only if X and Z are independent). In the limit as $\alpha \to 1$, $H_\alpha(\cdot)$ approaches the Shannon entropy (not an obvious fact), so $I_\alpha(\cdot)$ reduces to the regular mutual information. Similarly, we define

$$I_\alpha(X; Z|W) = \sum_w P_W(w) I_\alpha(X; Z|W = w),$$

where $I_\alpha(X; Z|W = w)$ is computed via (1) using the conditional distribution of X and Z given W.

Except in trivial cases, $H_\alpha(\cdot)$ is *strictly larger* than $H(\cdot)$ when $0 < \alpha < 1$; this means that the effects of extreme sparsity (few words per document, or too few occurrences of non-frequent words) on the estimation of entropy and mutual information can be dampened with an appropriate choice of α. This happens because extreme sparsity in the data yields empirical distributions which lie at, or close to, the boundary of the probability simplex. The entropy of such distributions is usually underestimated, compared to the smooth distributions which generate the data. Rényi's entropy is larger than Shannon's entropy, especially in those regions close to the boundary, and can thus provide an estimate which is closer to the true entropy.

3.2 Csiszár's Mutual Information

Csiszár defined the mutual information of order α as

$$I_\alpha^C(X; Z) = \min_Q \sum_z P_Z(z) D_\alpha(P_{X|Z}(\cdot|z) \| Q(\cdot)), \tag{3}$$

where $D_\alpha(\cdot\|\cdot)$ is the Rényi divergence (Csiszár, 1995). It was shown that $I_\alpha^C(X; Z)$ retains most of the properties of $I(X; Z)$—it is a non-negative, continuous, and concave function of P_X, it is convex in $P_{X|Z}$ for $\alpha < 1$, and converges to $I(X; Z)$ as $\alpha \to 1$.

Notably, $I_\alpha^C(X; Z) \leq I(X; Z)$ for $0 < \alpha < 1$; this means, as above, that α regulates the overestimation of mutual information that may result from data sparseness.

There is no analytic form for the minimizer of the right-hand-side of (3) (Csiszár, 1995), but it may be computed via an alternating minimization algorithm (Karakos et al., 2007a).

4 Experimental Methods and Results

We demonstrate the feasibility of cross-instance tuning with experiments on unsupervised document categorization from the 20 Newsgroups corpus (Lang, 1995); this corpus consists of roughly 20,000 news articles, evenly divided among 20 Usenet groups.

Random samples of 500 articles each were chosen by (Slonim et al., 2002) to create multiple test collections: 250 each from 2 arbitrarily chosen Usenet groups for the *Binary* test collection, 100 articles each from 5 groups for the *Multi5* test collection, and 50 each from 10 groups for the *Multi10* test collection. Three independent test collections of each kind (*Binary*, *Multi5* and *Multi10*) were created, for a total of 9 collections. The sIB method was used to separately cluster each collection, given the correct number of clusters.

A comparison of sIB and IDTs on the *same* 9 test collections was reported in (Karakos et al., 2007b; Karakos et al., 2007a). Matlab code from (Slonim, 2003) was used for the sIB experiments, while the parameterized mutual information measures of Section 3 were used for the IDTs. A comparison was also made with the EM-based Gaussian mixtures clustering tool *mclust* (Fraley and Raftery, 1999), and with a simple K-means algorithm. Since the two latter techniques gave uniformly worse clusterings than those of sIB and IDTs, we omit them from the following discussion.

To show that our methods work beyond the 9 particular 500-document collections described above, in this paper we instead use five *different* randomly sampled test collections for each of the *Binary*, *Multi5* and *Multi10* cases, making for a total of 15 new test collections in this paper. For diversity, we ensure that none of the five test collections (in each case) contain any documents used in the three collections of (Slonim et al., 2002) (for the same case).

We pre-process the documents of each test collection using the procedure[2] mentioned in (Karakos et al., 2007b). The 15 test collections are then converted to feature matrices—term-document frequency matrices for sIB, and discounted tf/idf matrices (according to the Okapi formula (Gatford et al., 1995)) for IDTs—with each row of a matrix representing one document in that test collection.

[2]Excluding the subject line, the header of each abstract is removed. Stop-words such as *a, the, is,* etc. are removed, and stemming is performed (e.g., common suffixes such as -ing, -er, -ed, etc., are removed). Also, all numbers are collapsed to one symbol, and non-alphanumeric sequences are converted to whitespace. Moreover, as suggested in (Yang and Pedersen, 1997) as an effective method for reducing the dimensionality of the feature space (number of distinct words), all words which occur fewer than t times in the corpus are removed. For the sIB experiments, we use $t = 2$ (as was done in (Slonim et al., 2002)), while for the IDT experiments we use $t = 3$; these choices result in the best performance for each method, respectively, on another dataset.

4.1 Selecting α with "Strapping"

In order to pick the value of the parameter α for each of the sIB and IDT test experiments, we use "strapping" (Eisner and Karakos, 2005), which, as we mentioned earlier, is a technique for training a meta-classifier that chooses among possible clusterings. The training is based on unrelated instances of the same clustering task. The final choice of clustering is still unsupervised, since no labels (or ground truth, in general) for the instance of interest are used.

Here, our collection of possible clusterings for each test collection is generated by varying the α parameter. Strapping does not care, however, how the collection was generated. (In the original strapping paper, for example, Eisner and Karakos (2005) generated their collection by bootstrapping word-sense classifiers from 200 different seeds.)

Here is how we choose a particular unsupervised α-clustering to output for a given test collection:

- We cluster the test collection (e.g., the first Multi5 collection) with various values of α, namely $\alpha = 0.1, 0.2, \ldots, 1.0$.
- We compute a feature vector from each of the clusterings. Note that the features are computed from only the clusterings and the data points, since no labels are available.
- Based on the feature vectors, we predict the "goodness" of each clustering, and return the "best" one.

How do we predict the "goodness" of a clustering? By first learning to distinguish good clusterings from bad ones, by using unrelated instances of the task on which we know the true labels:

- We cluster some unrelated datasets with various values of α, just as we will do in the test condition.
- We evaluate each of the resulting clusterings using the true labels on its dataset.[3]
- We train a "meta-classifier" that predicts the true rank (or accuracy) of each clustering based on the feature vector of the clustering.

[3]To evaluate a clustering, one only really needs the true labels on a *sample* of the dataset, although in our experiments we did have true labels on the entire dataset.

Specifically, for each task (Binary, Multi5, and Multi10) and each clustering method (sIB and IDT), a meta-classifier is learned thus:

- We obtain 10 clusterings ($\alpha = 0.1, 0.2, \ldots, 1.0$) for each of 5 unrelated task instances (datasets whose construction is described below).

- For each of these 50 clusterings, we compute the following 14 features: (i) One minus the average cosine of the angle (in tf/idf space) between each example and the centroid of the cluster to which it belongs. (ii) The average Rényi divergence, computed for parameters $1.0, 0.5, 0.1$, between the empirical distribution of each example and the centroid of the cluster to which it belongs. (iii) We create 10 more features, one per α. For the α used in this clustering, the feature value is equal to $e^{-0.1\bar{r}}$, where $\bar{r}$ is the average rank of the clustering (i.e., the average of the 4 ranks resulting from sorting all 10 clusterings (per training example) according to one of the 4 features in (i) and (ii)). For all other α's, the feature is set to zero. Thus, only α's which yield relatively good rankings can have non-zero features in the model.

- We normalize each group of 10 feature vectors, translating and scaling each of the 14 dimensions to make it range from 0 to 1. (We will do the same at test time.)

- We train ranking SVMs (Joachims, 2002), with a Gaussian kernel, to learn how to rank these 50 clusterings given their respective normalized feature vectors. The values of c, γ (which control regularization and the Gaussian kernel) were optimized through leave-one-out cross validation in order to maximize the average accuracy of the top-ranked clustering, over the 5 training sets. Once a local maximum of the average accuracy was obtained, further tuning of c, γ to maximize the Spearman rank correlation between the predicted and true ranks was performed.

A model trained in this way knows something about the task, and may work well for many new, unseen instances of the task. However, we presume that it will work best on a given test instance if trained on similar instances. The ideal would be to match the test collection in every aspect: (i) the number of training labels should be equal to the number of desired clusters of the test collection; (ii) the training clusters should be topically similar to the desired test clusters.

In our scenario, we enjoy the luxury of plenty of labeled data that can be used to create similar instances. Thus, given a test collection $\mathcal{A}$ to be clustered into L clusters, we create similar training sets by identifying the L training newsgroups whose centroids in tf/idf space (using the Okapi formula mentioned earlier) have the smallest angle to the centroid of $\mathcal{A}$.[4] (Of course, we exclude newsgroups that appear in $\mathcal{A}$.) We then form a supervised 500-document training set $\mathcal{A}'$ by randomly choosing $500/L$ documents from each of these L newsgroups; we do this 5 times to obtain 5 supervised training sets.

Table 1 shows averaged classification errors resulting from strapping (*"str"* rows) for the Jensen-Rényi divergence and Csiszár's mutual information, used within IDTs and sIB, respectively. (We also tried the reverse, using Jensen-Rényi in sIB and Csiszár's in IDTs, but the results were uniformly worse in the former case and no better in the latter case.) The "MI" rows show the classification errors of the untuned algorithms ($\alpha = 1$), which, in almost all cases, are worse than the tuned ones.

4.2 Tuning α on Statistically Similar Examples

We now show that strapping outperforms a simpler and more obvious method for cross-instance tuning. To cluster a test collection $\mathcal{A}$, we could simply tune the clustering algorithm by choosing the α that works best on a related task instance.

We again take care to construct a training instance $\mathcal{A}'$ that is closely related to the test instance $\mathcal{A}$. In fact, we take even greater care this time. Given $\mathcal{A}$,

[4] For each of the Binary collections, the closest training newsgroups in our experiments were *talk.politics.guns, talk.religion.misc*; for each of the Multi5 collections the closest newsgroups were *sci.electronics, rec.autos, sci.med, talk.politics.misc, talk.religion.misc*, and for the Multi10 collections they were *talk.politics.misc, rec.motorcycles, talk.religion.misc, comp.graphics, comp.sys.ibm.pc.hardware, rec.sport.baseball, comp.os.ms-windows.misc, comp.windows.x, soc.religion.christian, talk.politics.mideast.* Note that each of the Binary test collections happens to be closest to the *same* two training newsgroups; a similar behavior was observed for the Multi5 and Multi10 newsgroups.

	Method \ Set	Binary	Multi5	Multi10
IDTs	MI	11.3%	9.9%	42.2%
	I_α *(str)*	**10.4%**	**9.2%**	**39.0%**
	I_α *(rls)*	**10.1%**	10.4%	42.7%
sIB	MI	12.0%	6.8%	38.5%
	I_α^C *(str)*	**11.2%**	6.9%	**35.8%**
	I_α^C *(rls)*	**11.1%**	7.4%	**37.4%**

Table 1: Average classification errors for IDTs and sIB, using strapping (*"str"* rows) and regularized least squares (*"rls"* rows) to pick α in Jensen-Rényi divergence and Csiszár's mutual information. Rows "MI" show the errors resulting from the *untuned* algorithms, which use the regular mutual information objective ($\alpha = 1$). Results which are better than the corresponding "MI" results are shown in **bold**.

we identify the same set of L closest newsgroups as described above. This time, however, we carefully select $|\mathcal{A}|/L$ documents from each newsgroup rather than randomly choosing $500/L$ of them. Specifically, for each test example (document) $X \in \mathcal{A}$, we add a similar training example X' into $\mathcal{A}'$, chosen as follows:

We associate each test example X to the most similar of the L training newsgroups, under a constraint that only $|\mathcal{A}|/L$ training examples may be associated to each newsgroup. To do this, we iterate through all pairs (X, G) where X is a test example and G is a training newsgroup, in increasing order by the angle between X and G. If X is not yet associated and G is not yet "full," then we associate X with G, and choose X' to be the document in G with the smallest angle to X.

We cluster $\mathcal{A}'$ 10 times, for $\alpha = 0.1, \ldots, 1.0$, and we collect supervised error results $\mathcal{E}(\alpha)$, $\alpha \in \{0.1, \ldots, 1.0\}$. Now, instead of using the single best $\alpha^* = \operatorname{argmin}_\alpha \mathcal{E}(\alpha)$ to cluster $\mathcal{A}$ (which may result in overfitting) we use regularized least-squares (RLS) (Hastie et al., 2001), where we try to approximate the *probability that an α is the best.* The estimated probabilities are given by

$$\hat{\boldsymbol{p}} = \mathbf{K}(\lambda\mathbf{I} + \mathbf{K})^{-1}\boldsymbol{p},$$

where $\mathbf{I}$ is the unit matrix, $\boldsymbol{p}$ is the training probability of the best α (i.e., it is 1 at the position of α^* and zero elsewhere), and $\mathbf{K}$ is the kernel matrix, where $K(i,j) = \exp(-(\mathcal{E}(\alpha_i) - \mathcal{E}(\alpha_j))^2/\sigma^2)$ is the value of the kernel which expresses the "similarity" between two clusterings of the same training dataset, in terms of their errors. The parameters σ, γ are set to $0.5, 0.1$, respectively, after performing a (local) maximization of the Spearman correlation between training accuracies and predicted probabilities $\hat{\boldsymbol{p}}$, for all 15 training instances. After performing a linear normalization of $\hat{\boldsymbol{p}}$ to make it a probability vector, the average predicted value of α, i.e., $\hat{\alpha} = \sum_{i=1}^{10} \hat{p}_i\, \alpha_i$, (rounded-off to one of $\{0.1, \ldots, 1.0\}$) is used to cluster $\mathcal{A}$.

Table 1 shows the average classification error results using RLS (*"rls"* rows). We can see that, on average over the 15 test instances, the error rate of the tuned IDTs and sIB algorithms is lower than that of the untuned algorithms, so cross-instance tuning was effective. On the other hand, the errors are generally higher than that of the strapping method, which examines the results of using different α values on $\mathcal{A}$.

5 Concluding Remarks

We have considered the problem of cross-instance tuning of two unsupervised document clustering algorithms, through the introduction of a degree of freedom into their mutual information objective. This degree of freedom is tuned using *labeled* document collections (which are unrelated to the test collections); we explored two approaches for performing the tuning: (i) through a judicious sampling of training data, to match the marginal statistics of the test data, and (ii) via "strapping", which trains a meta-classifier to distinguish between good and bad clusterings. Our unsupervised categorization experiments from the "20 Newsgroups" corpus indicate that, although both approaches improve the baseline algorithms, "strapping" is clearly a better choice for knowledge transfer between unrelated task instances.

References

R. K. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, Nov.

S. Ben-David and R. Schuller. 2003. Exploiting task relatedness for multiple task learning. In *Proc. of the Sixteenth Annual Conference on Learning Theory (COLT-03)*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory (COLT-98)*, pages 92–100.

R. Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

T. Cover and J. Thomas. 1991. *Elements of Information Theory*. John Wiley and Sons.

I. Csiszár. 1995. Generalized cutoff rates and Rényi's information measures. *IEEE Trans. on Information Theory*, 41(1):26–34, January.

I. Dhillon, S. Mallela, and R. Kumar. 2003. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research (JMLR), Special Issue on Variable and Feature Selection*, pages 1265–1287, March.

J. Eisner and D. Karakos. 2005. Bootstrapping without the boot. In *Proc. 2005 Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, October.

T. Evgeniou and M. Pontil. 2004. Regularized multi-task learning. In *Proc. Knowledge Discovery and Data Mining*.

C. Fraley and A. E. Raftery. 1999. Mclust: Software for model-based cluster analysis. *Journal on Classification*, 16:297–306.

M. Gatford, M. M. Hancock-Beaulieu, S. Jones, S. Walker, and S. E. Robertson. 1995. Okapi at TREC-3. In *The Third Text Retrieval Conference (TREC-3)*, pages 109–126.

A. Ben Hamza and H. Krim. 2003. Jensen-Rényi divergence measure: Theoretical and computational perspectives. In *Proc. IEEE Int. Symp. on Information Theory*, Yokohama, Japan, June.

T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning*. Springer-Verlag.

A. O. Hero, B. Ma, O. Michel, and J. Gorman. 2001. Alpha-divergence for classification, indexing and retrieval. Technical Report CSPL-328, University of Michigan Ann Arbor, Communications and Signal Processing Laboratory, May.

F. Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM Conf. on Knowledge Discovery and Data Mining (KDD)*.

D. Karakos, S. Khudanpur, J. Eisner, and C. E. Priebe. 2005. Unsupervised classification via decision trees: An information-theoretic perspective. In *Proc. 2005 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005)*, March.

D. Karakos, S. Khudanpur, J. Eisner, and C. E. Priebe. 2007a. Information-theoretic aspects of iterative denoising. Submitted to the Journal of Machine Learning Research, February.

D. Karakos, S. Khudanpur, J. Eisner, and C. E. Priebe. 2007b. Iterative denoising using Jensen-Rényi divergences with an application to unsupervised document categorization. In *Proc. 2007 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, April.

K. Lang. 1995. Learning to filter netnews. In *Proc. 13th Int. Conf. on Machine Learning*, pages 331–339.

David D. Lewis and Philip J. Hayes. 1994. Guest editorial. *ACM Transactions on Information Systems*, 12(3):231, July.

C. E. Priebe, D. J. Marchette, and D. M. Healy. 2004a. Integrated sensing and processing decision trees. *IEEE Trans. on Pat. Anal. and Mach. Intel.*, 26(6):699–708, June.

C. E. Priebe, D. J. Marchette, Y. Park, E. Wegman, J. Solka, D. Socolinsky, D. Karakos, K. Church, R. Guglielmi, R. Coifman, D. Lin, D. Healy, M. Jacobs, and A. Tsao. 2004b. Iterative denoising for cross-corpus discovery. In *Proc. 2004 International Symposium on Computational Statistics (COMPSTAT 2004)*, August.

N. Slonim and N. Tishby. 2000. Document clustering using word clusters via the information bottleneck method. In *Research and Development in Information Retrieval*, pages 208–215.

N. Slonim, N. Friedman, and N. Tishby. 2002. Unsupervised document classification using sequential information maximization. In *Proc. SIGIR'02, 25th ACM Int. Conf. on Research and Development of Inform. Retrieval*.

N. Slonim. 2003. IBA_1.0: Matlab code for information bottleneck clustering algorithms. Available from http://www.princeton.edu/∼nslonim/IB_Release1.0/IB_Release1_0.tar.

N. Tishby, F. Pereira, and W. Bialek. 1999. The information bottleneck method. In *37th Allerton Conference on Communication and Computation*.

K. Torkkola. 2002. On feature extraction by mutual information maximization. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP-2002)*, May.

Y. Yang and J. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Intl. Conf. on Machine Learning (ICML-97)*, pages 412–420.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.

Using "Annotator Rationales" to Improve Machine Learning for Text Categorization[*]

Omar F. Zaidan and **Jason Eisner**
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA
{ozaidan,jason}@cs.jhu.edu

Christine D. Piatko
JHU Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723 USA
christine.piatko@jhuapl.edu

Abstract

We propose a new framework for supervised machine learning. Our goal is to learn from smaller amounts of supervised training data, by collecting a richer *kind* of training data: annotations with "rationales." When annotating an example, the human teacher will also highlight evidence supporting this annotation—thereby teaching the machine learner *why* the example belongs to the category. We provide some rationale-annotated data and present a learning method that exploits the rationales during training to boost performance significantly on a sample task, namely sentiment classification of movie reviews. We hypothesize that in some situations, providing rationales is a more fruitful use of an annotator's time than annotating more examples.

1 Introduction

Annotation cost is a bottleneck for many natural language processing applications. While supervised machine learning systems are effective, it is labor-intensive and expensive to construct the many training examples needed. Previous research has explored active or semi-supervised learning as possible ways to lessen this burden.

We propose a new way of breaking this annotation bottleneck. Annotators currently indicate *what* the correct answers are on training data. We propose that they should also indicate *why*, at least by coarse hints. We suggest new machine learning approaches that can benefit from this "why" information.

For example, an annotator who is categorizing phrases or documents might also be asked to highlight a few substrings that significantly influenced her judgment. We call such clues "rationales." They need not correspond to machine learning features.

[*]This work was supported by the JHU WSE/APL Partnership Fund; National Science Foundation grant No. 0347822 to the second author; and an APL Hafstad Fellowship to the third.

In some circumstances, rationales should not be too expensive or time-consuming to collect. As long as the annotator is spending the time to study example x_i and classify it, it may not require much extra effort for her to mark reasons for her classification.

2 Using Rationales to Aid Learning

We will not rely exclusively on the rationales, but use them only as an added source of information. The idea is to help direct the learning algorithm's attention—helping it tease apart signal from noise.

Machine learning algorithms face a well-known "credit assignment" problem. Given a complex datum x_i and the desired response y_i, many features of x_i could be responsible for the choice of y_i. The learning algorithm must tease out which features were *actually* responsible. This requires a lot of training data, and often a lot of computation as well.

Our rationales offer a shortcut to solving this "credit assignment" problem, by providing the learning algorithm with hints as to which features of x_i were relevant. Rationales should help guide the learning algorithm toward the correct classification function, by pushing it toward a function that correctly pays attention to each example's relevant features. This should help the algorithm learn from less data and avoid getting trapped in local maxima.[1]

In this paper, we demonstrate the "annotator rationales" technique on a text categorization problem previously studied by others.

[1]To understand the local maximum issue, consider the hard problem of training a standard 3-layer feed-forward neural network. If the activations of the "hidden" layer's features (nodes) were observed at training time, then the network would decompose into a pair of independent 2-layer perceptrons. This turns an NP-hard problem with local maxima (Blum and Rivest, 1992) to a polytime-solvable convex problem. Although rationales might only provide *indirect* evidence of the hidden layer, this would still modify the objective function (see section 8) in a way that tended to make the correct weights easier to discover.

3 Discriminative Approach

One popular approach for text categorization is to use a discriminative model such as a Support Vector Machine (SVM) (e.g. (Joachims, 1998; Dumais, 1998)). We propose that SVM training can in general incorporate annotator rationales as follows.

From the rationale annotations on a positive example $\overrightarrow{x_i}$, we will construct one or more "not-quite-as-positive" *contrast examples* $\overrightarrow{v_{ij}}$. In our text categorization experiments below, each contrast document $\overrightarrow{v_{ij}}$ was obtained by starting with the original and "masking out" one or all of the several rationale substrings that the annotator had highlighted (r_{ij}). The intuition is that the *correct* model should be less sure of a positive classification on the contrast example $\overrightarrow{v_{ij}}$ than on the original example $\vec{x_i}$, because $\overrightarrow{v_{ij}}$ lacks evidence that the annotator found significant.

We can translate this intuition into additional constraints on the correct model, i.e., on the weight vector $\vec{w}$. In addition to the usual SVM constraint on positive examples that $\vec{w} \cdot \overrightarrow{x_i} \geq 1$, we also want (for each j) that $\vec{w} \cdot \vec{x}_i - \vec{w} \cdot \overrightarrow{v_{ij}} \geq \mu$, where $\mu \geq 0$ controls the size of the desired margin between original and contrast examples.

An ordinary soft-margin SVM chooses $\vec{w}$ and $\vec{\xi}$ to minimize

$$\frac{1}{2}\|\vec{w}\|^2 + C(\sum_i \xi_i) \tag{1}$$

subject to the constraints

$$(\forall i) \;\; \vec{w} \cdot \overrightarrow{x_i} \cdot y_i \;\; \geq \;\; 1 - \xi_i \tag{2}$$

$$(\forall i) \;\; \xi_i \;\; \geq \;\; 0 \tag{3}$$

where $\overrightarrow{x_i}$ is a training example, $y_i \in \{-1, +1\}$ is its desired classification, and ξ_i is a slack variable that allows training example $\overrightarrow{x_i}$ to miss satisfying the margin constraint if necessary. The parameter $C > 0$ controls the cost of taking such slack, and should generally be lower for noisier or less linearly separable datasets. We add the *contrast constraints*

$$(\forall i, j) \;\; \vec{w} \cdot (\overrightarrow{x_i} - \overrightarrow{v_{ij}}) \cdot y_i \geq \mu(1 - \xi_{ij}), \tag{4}$$

where $\overrightarrow{v_{ij}}$ is one of the contrast examples constructed from example $\overrightarrow{x_i}$, and $\xi_{ij} \geq 0$ is an associated slack variable. Just as these extra constraints have their own margin μ, their slack variables have their own cost, so the objective function (1) becomes

$$\frac{1}{2}\|\vec{w}\|^2 + C(\sum_i \xi_i) + C_{contrast}(\sum_{i,j} \xi_{ij}) \tag{5}$$

The parameter $C_{contrast} \geq 0$ determines the importance of satisfying the contrast constraints. It should generally be less than C if the contrasts are noisier than the training examples.[2]

In practice, it is possible to solve this optimization using a standard soft-margin SVM learner. Dividing equation (4) through by μ, it becomes

$$(\forall i, j) \;\; \vec{w} \cdot \overrightarrow{x_{ij}} \cdot y_i \geq 1 - \xi_{ij}, \tag{6}$$

where $\overrightarrow{x_{ij}} \overset{\text{def}}{=} \frac{\overrightarrow{x_i} - \overrightarrow{v_{ij}}}{\mu}$. Since equation (6) takes the same form as equation (2), we simply add the pairs $(\overrightarrow{x_{ij}}, y_i)$ to the training set as *pseudoexamples*, weighted by $C_{contrast}$ rather than C so that the learner will use the objective function (5).

There is one subtlety. To allow a biased hyperplane, we use the usual trick of prepending a 1 element to each training example. Thus we require $\vec{w} \cdot (1, \overrightarrow{x_i}) \geq 1 - \xi_i$ (which makes w_0 play the role of a bias term). This means, however, that we must prepend a 0 element to each pseudoexample: $\vec{w} \cdot \frac{(1, \vec{x}_i) - (1, \overrightarrow{v_{ij}})}{\mu} = \vec{w} \cdot (0, \overrightarrow{x_{ij}}) \geq 1 - \xi_{ij}$.

In our experiments, we optimize μ, C, and $C_{contrast}$ on held-out data (see section 5.2).

4 Rationale Annotation for Movie Reviews

In order to demonstrate that annotator rationales help machine learning, we needed annotated data that included rationales for the annotations.

We chose a dataset that would be enjoyable to re-annotate: the movie review dataset of (Pang et al., 2002; Pang and Lee, 2004).[3] The dataset consists of 1000 positive and 1000 negative movie reviews obtained from the Internet Movie Database (IMDb) review archive, all written before 2002 by a total of 312 authors, with a cap of 20 reviews per author per

[2]Taking $C_{contrast}$ to be constant means that all rationales are equally valuable. One might instead choose, for example, to reduce $C_{contrast}$ for examples x_i that have *many* rationales, to prevent x_i's contrast examples v_{ij} from together dominating the optimization. However, in this paper we assume that an x_i with more rationales really does provide more evidence about the true classifier $\vec{w}$.

[3]Polarity dataset version 2.0.

category. Pang and Lee have divided the 2000 documents into 10 folds, each consisting of 100 positive reviews and 100 negative reviews.

The dataset is arguably artificial in that it keeps only reviews where the reviewer provided a rather high or rather low numerical rating, allowing Pang and Lee to designate the review as positive or negative. Nonetheless, most reviews contain a difficult mix of praise, criticism, and factual description. In fact, it is possible for a mostly critical review to give a positive overall recommendation, or vice versa.

4.1 Annotation procedure

Rationale annotators were given guidelines[4] that read, in part:

Each review was intended to give either a positive or a negative overall recommendation. You will be asked to justify why a review is positive or negative. To justify why a review is positive, highlight the most important words and phrases that would tell someone to see the movie. To justify why a review is negative, highlight words and phrases that would tell someone not to see the movie. These words and phrases are called **rationales**.

You can highlight the rationales as you notice them, which should result in several rationales per review. Do your best to mark enough rationales to provide convincing support for the class of interest.

You do not need to go out of your way to mark everything. You are probably doing too much work if you find yourself going back to a paragraph to look for even more rationales in it. Furthermore, it is perfectly acceptable to skim through sections that you feel would not contain many rationales, such as a reviewer's plot summary, even if that might cause you to miss a rationale here and there.

The last two paragraphs were intended to provide some guidance on how *many* rationales to annotate. Even so, as section 4.2 shows, some annotators were considerably more thorough (and slower).

Annotators were also shown the following examples[5] of positive rationales:

- **you will enjoy the hell out of** American Pie.
- fortunately, they **managed to do it in an interesting and funny way**.
- he is **one of the most exciting martial artists on the big screen**, continuing to perform his own stunts and **dazzling audiences** with his flashy kicks and punches.
- the romance was **enchanting**.

and the following examples[5] of negative rationales:

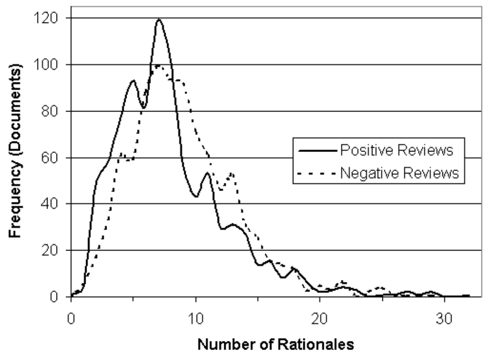


Figure 1: Histograms of rationale counts per document (A0's annotations). The overall mean of 8.55 is close to that of the four annotators in Table 1. The median and mode are 8 and 7.

- A woman in peril. A confrontation. An explosion. The end. **Yawn. Yawn. Yawn.**
- when a film makes watching Eddie Murphy **a tedious experience, you know something is terribly wrong**.
- the movie is **so badly put together** that even the most casual viewer may notice the **miserable pacing and stray plot threads**.
- **don't go see** this movie

The annotation involves **boldfacing** the rationale phrases using an HTML editor. Note that a fancier annotation tool would be necessary for a task like named entity tagging, where an annotator must mark many named entities in a single document. At any given moment, such a tool should allow the annotator to highlight, view, and edit only the several rationales for the "current" annotated entity (the one most recently annotated or re-selected).

One of the authors (A0) annotated folds 0–8 of the movie review set (1,800 documents) with rationales that supported the gold-standard classifications. This training/development set was used for all of the learning experiments in sections 5–6. A histogram of rationale counts is shown in Figure 1. As mentioned in section 3, the rationale annotations were just textual substrings. The annotator did not require knowledge of the classifier features. Thus, our rationale dataset is a new resource[4] that could also be used to study exploitation of rationales under feature sets or learning methods other than those considered here (see section 8).

4.2 Inter-annotator agreement

To study the annotation process, we randomly selected 150 documents from the dataset. The doc-

[4]Available at http://cs.jhu.edu/∼ozaidan/rationales.

[5]For our controlled study of annotation time (section 4.2), different examples were given with full document context.

	Rationales per document	% rationales also annotated by A1	% rationales also annotated by A2	% rationales also annotated by AX	% rationales also annotated by **AY**	% rationales also ann. by **anyone else**
A1	5.02	(100)	69.6	63.0	80.1	91.4
A2	10.14	42.3	(100)	50.2	67.8	80.9
AX	6.52	49.0	68.0	(100)	79.9	90.9
AY	11.36	39.7	56.2	49.3	(100)	75.5

Table 1: Average number of rationales and inter-annotator agreement for Tasks 2 and 3. A rationale by Ai ("I think **this is a great** movie!") is considered to have been annotated also by Aj if at least one of Aj's rationales overlaps it ("I think this is a **great movie!**"). In computing pairwise agreement on rationales, we ignored documents where Ai and Aj disagreed on the class. Notice that the most thorough annotator **AY** caught most rationales marked by the others (exhibiting high "recall"), and that most rationales enjoyed some degree of consensus, especially those marked by the least thorough annotator **A1** (exhibiting high "precision").

uments were split into three groups, each consisting of 50 documents (25 positive and 25 negative). Each subset was used for one of three tasks:[6]

- **Task 1:** Given the document, annotate only the class (positive/negative).
- **Task 2:** Given the document and its class, annotate some rationales for that class.
- **Task 3:** Given the document, annotate both the class and some rationales for it.

We carried out a pilot study (annotators AX and AY: two of the authors) and a later, more controlled study (annotators A1 and A2: paid students). The latter was conducted in a more controlled environment where both annotators used the same annotation tool and annotation setup as each other. Their guidelines were also more detailed (see section 4.1). In addition, the documents for the different tasks were interleaved to avoid any practice effect.

The annotators' classification accuracies in Tasks 1 and 3 (against Pang & Lee's labels) ranged from 92%–97%, with 4-way agreement on the class for 89% of the documents, and pairwise agreement also ranging from 92%–97%. Table 1 shows how many rationales the annotators provided and how well their rationales agreed.

Interestingly, in Task 3, four of **AX's rationales for a positive class** were also partially highlighted by AY as support for AY's (incorrect) *negative* classifications, such as:

[6]Each task also had a "warmup" set of 10 documents to be annotated before that tasks's 50 documents. Documents for Tasks 2 and 3 would automatically open in an HTML editor while Task 1 documents opened in an HTML viewer with no editing option. The annotators recorded their classifications for Tasks 1 and 3 on a spreadsheet.

min./KB	A1 time	A2 time	AX time	AY time
Task 1	0.252	0.112	0.150	0.422
Task 2	0.396	0.537	0.242	0.626
Task 3	0.399	0.505	0.288	1.01
min./doc.	A1 time	A2 time	AX time	AY time
Task 1	1.04	0.460	0.612	1.73
min./rat.	A1 time	A2 time	AX time	AY time
Task 2	0.340	0.239	0.179	0.298
Task 3	0.333	0.198	0.166	0.302

Table 2: Average annotation rates on each task.

- **Even with its numerous flaws, the movie all comes together**, if only for those who ...
- "Beloved" acts like an incredibly difficult **chamber drama paired with a ghost story**.

4.3 Annotation time

Average annotation times are in Table 2. As hoped, rationales did not take too much extra time for most annotators to provide. For each annotator except A2, providing rationales only took roughly twice the time (Task 3 vs. Task 1), even though it meant marking an average of 5–11 rationales in addition to the class.

Why this low overhead? Because marking the class already required the Task 1 annotator to read the document and find some rationales, even if s/he did not mark them. The only extra work in Task 3 is in making them explicit. This synergy between class annotation and rationale annotation is demonstrated by the fact that doing both at once (Task 3) was faster than doing them separately (Tasks 1+2).

We remark that this task—binary classification on full documents—seems to be almost a worst-case scenario for the annotation of rationales. At a purely mechanical level, it was rather heroic of A0 to attach 8–9 new rationale phrases r_{ij} to every bit y_i of ordinary annotation. Imagine by contrast a more local task of identifying entities or relations. Each lower-level annotation y_i will tend to have fewer rationales r_{ij}, while y_i itself will be more complex and hence more difficult to mark. Thus, we expect that the overhead of collecting rationales will be less in many scenarios than the factor of 2 we measured.

Annotation overhead could be further reduced. For a multi-class problem like relation detection, one could ask the annotator to provide rationales *only* for the rarer classes. This small amount of extra time where the data is sparsest would provide extra guidance where it was most needed. Another possibility is passive collection of rationales via eye tracking.

5 Experimental Procedures

5.1 Feature extraction

Although this dataset seems to demand discourse-level features that contextualize bits of praise and criticism, we exactly follow Pang et al. (2002) and Pang and Lee (2004) in merely using binary unigram features, corresponding to the 17,744 unstemmed word or punctuation types with count ≥ 4 in the full 2000-document corpus. Thus, each document is reduced to a 0-1 vector with 17,744 dimensions, which is then normalized to unit length.[7]

We used the method of section 3 to place additional constraints on a linear classifier. Given a training document, we create several contrast documents, each by deleting exactly one rationale substring from the training document. Converting documents to feature vectors, we obtained an original example $\overrightarrow{x_i}$ and several contrast examples $\overrightarrow{v_{i1}}, \overrightarrow{v_{i2}}, \ldots$.[8] Again, our training method required each original document to be classified more confidently (by a margin μ) than its contrast documents.

If we were using more than unigram features, then simply *deleting* a rationale substring would not always be the best way to create a contrast document, as the resulting ungrammatical sentences might cause deep feature extraction to behave strangely (e.g., parse errors during preprocessing). The goal in creating the contrast document is merely to suppress features (n-grams, parts of speech, syntactic dependencies ...) that depend in part on material in one or more rationales. This could be done directly by modifying the feature extractors, or if one prefers to use existing feature extractors, by "masking" rather than deleting the rationale substring—e.g., replacing each of its word tokens with a special MASK token that is treated as an out-of-vocabulary word.

5.2 Training and testing procedures

We transformed this problem to an SVM problem (see section 3) and applied SVM^{light} for training and testing, using the default linear kernel. We used only A0's rationales and the true classifications.

Fold 9 was reserved as a test set. All accuracy results reported in the paper are the result of testing on fold 9, after training on subsets of folds 0–8.

Our learning curves show accuracy after training on $T < 9$ folds (i.e., $200T$ documents), for various T. To reduce the noise in these results, the accuracy we report for training on T folds is actually the average of 9 different experiments with different (albeit overlapping) training sets that cover folds 0–8:

$$\frac{1}{9}\sum_{i=0}^{8} acc(F_9 \mid \theta^*, F_{i+1} \cup \ldots \cup F_{i+T}) \qquad (7)$$

where F_j denotes the fold numbered $j \bmod 9$, and $acc(Z \mid \theta, Y)$ means classification accuracy on the set Z after training on Y with hyperparameters θ.

To evaluate whether two different training methods A and B gave significantly different average-accuracy values, we used a paired permutation test (generalizing a sign test). The test assumes independence among the 200 test examples but not among the 9 overlapping training sets. For each of the 200 test examples in fold 9, we measured (a_i, b_i), where a_i (respectively b_i) is the number of the 9 training sets under which A (respectively B) classified the example correctly. The p value is the probability that the absolute difference between the average-accuracy values would reach or exceed the observed absolute difference, namely $|\frac{1}{200}\sum_{i=1}^{200}\frac{a_i-b_i}{9}|$, if each (a_i, b_i) had an independent 1/2 chance of being replaced with (b_i, a_i), as per the null hypothesis that A and B are indistinguishable.

For any given value of T and any given training method, we chose hyperparameters $\theta^* =$

[7]The vectors are normalized *before* prepending the 1 corresponding to the bias term feature (mentioned in section 3).

[8]The contrast examples were not normalized to precisely unit length, but instead were normalized by the same factor used to normalize $\overrightarrow{x_i}$. This conveniently ensured that the pseudoexamples $\overrightarrow{x_{ij}} \overset{\text{def}}{=} \frac{\vec{x}_i - \overrightarrow{v_{ij}}}{\mu}$ were sparse vectors, with 0 coordinates for all words not in the jth rationale.

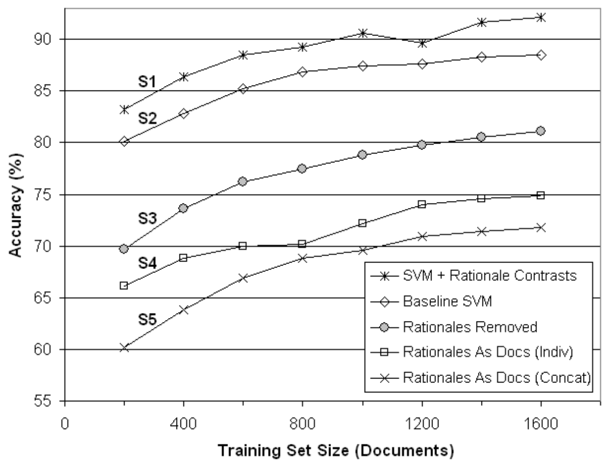


Figure 2: Classification accuracy under five different experimental setups (S1–S5). At each training size, the 5 accuracies are pairwise significantly different (paired permutation test, $p < 0.02$; see section 5.2), except for {S3,S4} or {S4,S5} at some sizes.

$(C, \mu, C_{contrast})$ to maximize the following cross-validation performance:[9]

$$\theta^* = \operatorname*{argmax}_{\theta} \sum_{i=0}^{8} acc(F_i \mid \theta, F_{i+1} \cup \ldots \cup F_{i+T}) \tag{8}$$

We used a simple alternating optimization procedure that begins at $\theta_0 = (1.0, 1.0, 1.0)$ and cycles repeatedly through the three dimensions, optimizing along each dimension by a local grid search with resolution 0.1.[10] Of course, when training without rationales, we did not have to optimize μ or $C_{contrast}$.

6 Experimental Results

6.1 The value of rationales

The top curve (S1) in Figure 2 shows that performance does increase when we introduce rationales for the training examples as contrast examples (section 3). S1 is significantly higher than the baseline curve (S2) immediately below it, which trains an ordinary SVM classifier without using rationales. At the largest training set size, rationales raise the accuracy from 88.5% to 92.2%, a 32% error reduction.

[9]One might obtain better performance (across *all* methods being compared) by choosing a separate θ^* for each of the 9 training sets. However, to simulate real limited-data training conditions, one should then find the θ^* for each $\{i, ..., j\}$ using a separate cross-validation within $\{i, ..., j\}$ only; this would slow down the experiments considerably.

[10]For optimizing along the C dimension, one could use the efficient method of Beineke et al. (2004), but not in SVM^{light}.

The lower three curves (S3–S5) show that learning is separately helped by the rationale and the non-rationale portions of the documents. S3–S5 are degraded versions of the baseline S2: they are ordinary SVM classifiers that perform significantly worse than S2 ($p < 0.001$).

Removing the rationale phrases from the training documents (S3) made the test documents much harder to discriminate (compared to S2). This suggests that annotator A0's rationales often covered *most* of the usable evidence for the true class.

However, the pieces to solving the classification puzzle cannot be found solely in the short rationale phrases. Removing all *non*-rationale text from the training documents (S5) was even worse than removing the rationales (S3). In other words, we cannot hope to do well simply by training on just the rationales (S5), although that approach is improved somewhat in S4 by treating each rationale (similarly to S1) as a *separate* SVM training example.

This presents some insight into why our method gives the best performance. The classifier in S1 is able to extract subtle patterns from the corpus, like S2, S3, or any other standard machine learning method, but it is *also* able to learn from a human annotator's decision-making strategy.

6.2 Using fewer rationales

In practice, one might annotate rationales for only *some* training documents—either when annotating a new corpus or when adding rationales *post hoc* to an existing corpus. Thus, a range of options can be found between curves S2 and S1 of Figure 2.

Figure 3 explores this space, showing how far the learning curve S2 moves upward if one has time to annotate rationales for a fixed number of documents R. The key useful discovery is that much of the benefit can actually be obtained with relatively few rationales. For example, with 800 training documents, annotating (0%, 50%, 100%) of them with rationales gives accuracies of (86.9%, 89.2%, 89.3%). With the maximum of 1600 training documents, annotating (0%, 50%, 100%) with rationales gives (88.5%, 91.7%, 92.2%).

To make this point more broadly, we find that the $R = 200$ curve is significantly above the $R = 0$ curve ($p < 0.05$) at all $T \leq 1200$. By contrast, the $R = 800, R = 1000, \ldots R = 1600$ points at each T

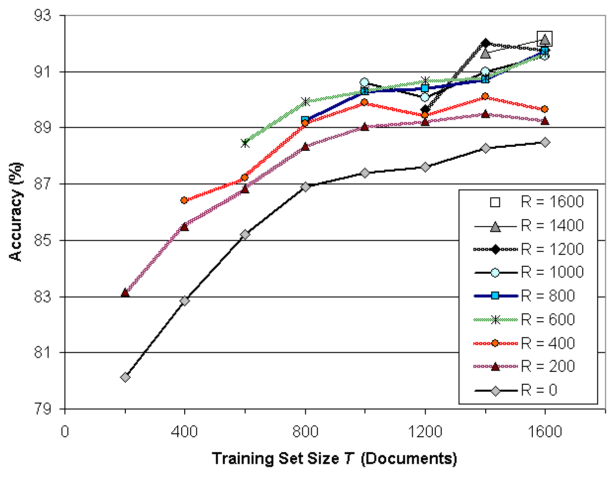


Figure 3: Classification accuracy for $T \in \{200, 400, ..., 1600\}$ training documents (x-axis) when only $R \in \{0, 200, ..., T\}$ of them are annotated with rationales (different curves). The $R = 0$ curve above corresponds to the baseline S2 from Figure 2. S1's points are found above as the leftmost points on the other curves, where $R = T$.

value are all-pairs statistically indistinguishable.

The figure also suggests that rationales and documents may be somewhat orthogonal in their benefit. When one has many documents and few rationales, there is no longer much benefit in adding more documents (the curve is flattening out), but adding more rationales seems to provide a fresh benefit: rationales have not yet reached *their* point of diminishing returns. (While this fresh benefit was often statistically significant, and greater than the benefit from more documents, our experiments did not establish that it was significantly greater.)

The above experiments keep *all* of A0's rationales on *a fraction of* training documents. We also experimented with keeping *a fraction of* A0's rationales (chosen randomly with randomized rounding) on *all* training documents. This yielded no noteworthy or statistically significant differences from Figure 3.

These latter experiments simulate a "lazy annotator" who is less assiduous than A0. Such annotators may be common in the real world. We also suspect that they will be more desirable. First, they should be able to add more rationales per hour than the A0-style annotator from Figure 3: some rationales are simply more noticeable than others, and a lazy annotator will quickly find the most noticeable ones without wasting time tracking down the rest. Second, the "most noticeable" rationales that they mark may be the most effective ones for learning, although our random simulation of laziness could not test that.

7 Related Work

Our rationales resemble "side information" in machine learning—supplementary information about the target function that is available at training time. Side information is sometimes encoded as "virtual examples" like our contrast examples or pseudoexamples. However, past work generates these by *automatically* transforming the training examples in ways that are expected to preserve or alter the classification (Abu-Mostafa, 1995). In another formulation, virtual examples are automatically generated but must be manually annotated (Kuusela and Ocone, 2004). Our approach differs because a human helps to generate the virtual examples. Enforcing a margin between ordinary examples and contrast examples also appears new.

Other researchers have considered how to reduce annotation effort. In active learning, the annotator classifies only documents where the system so far is less confident (Lewis and Gale, 1994), or in an information extraction setting, incrementally corrects details of the system's less confident entity segmentations and labelings (Culotta and McCallum, 2005).

Raghavan et al. (2005) asked annotators to identify globally "relevant" *features*. In contrast, our approach does not force the annotator to evaluate the importance of features individually, nor in a global context outside any specific document, nor even to know the learner's feature space. Annotators only mark text that supports their classification decision. Our methods then consider the combined effect of this text on the feature vector, which may include complex features not known to the annotator.

8 Future Work: Generative models

Our SVM contrast method (section 3) is not the only possible way to use rationales. We would like to explicitly *model* rationale annotation as a noisy process that reflects, imperfectly and incompletely, the annotator's internal decision procedure.

A natural approach would start with log-linear models in place of SVMs. We can define a probabilistic classifier

$$p_\theta(y \mid x) \overset{\text{def}}{=} \frac{1}{Z(x)} \exp \sum_{h=1}^{k} \theta_h f_h(x, y) \qquad (9)$$

where $\vec{f}(\cdot)$ extracts a feature vector from a classified document.

A standard training method would be to choose θ to maximize the conditional likelihood of the training classifications:

$$\underset{\vec{\theta}}{\operatorname{argmax}} \prod_{i=1}^{n} p_\theta(y_i \mid x_i) \tag{10}$$

When a rationale r_i is also available for each (x_i, y_i), we propose to maximize a likelihood that tries to predict these rationale data *as well*:

$$\underset{\vec{\theta}}{\operatorname{argmax}} \prod_{i=1}^{n} p_\theta(y_i \mid x_i) \cdot p_{\theta'}(r_i \mid x_i, y_i, \theta) \tag{11}$$

Notice that a given guess of θ might make equation (10) large, yet accord badly with the annotator's rationales. In that case, the second term of equation (11) will exert pressure on θ to change to something that conforms more closely to the rationales. If the annotator is correct, such a θ will generalize better beyond the training data.

In equation (11), $p_{\theta'}$ models the stochastic process of rationale annotation. What is an annotator actually doing when she annotates rationales? In particular, how do her rationales derive from the true value of θ and thereby tell us about θ? Building a good model $p_{\theta'}$ of rationale annotation will require some exploratory data analysis. Roughly, we expect that if $\theta_h f_h(x_i, y)$ is much higher for $y = y_i$ than for other values of y, then the annotator's r_i is correspondingly more likely to indicate in some way that feature f_h strongly influenced annotation y_i. However, we must also model the annotator's limited patience (she may not annotate all important features), sloppiness (she may indicate only indirectly that f_h is important), and bias (tendency to annotate some *kinds* of features at the expense of others).

One advantage of this generative approach is that it eliminates the need for contrast examples. Consider a non-textual example in which an annotator highlights the line crossing in a digital image of the digit "8" to mark the rationale that distinguishes it from "0." In this case it is not clear how to mask out that highlighted rationale to create a contrast example in which relevant features would not fire.[11]

9 Conclusions

We have proposed a quite simple approach to improving machine learning by exploiting the cleverness of annotators, asking them to provide enriched annotations for training. We developed and tested a particular discriminative method that can use "annotator rationales"—even on a fraction of the training set—to significantly improve sentiment classification of movie reviews.

We found fairly good annotator agreement on the rationales themselves. Most annotators provided several rationales per classification without taking too much extra time, even in our text classification scenario, where the rationales greatly outweigh the classifications in number and complexity. Greater speed might be possible through an improved user interface or passive feedback (e.g., eye tracking).

In principle, many machine learning methods might be modified to exploit rationale data. While our experiments in this paper used a discriminative SVM, we plan to explore generative approaches.

References

Y. S. Abu-Mostafa. 1995. Hints. *Neural Computation*, 7:639–671, July.

P. Beineke, T. Hastie, and S. Vaithyanathan. 2004. The sentimental factor: Improving review classification via human-provided information. In *Proc. of ACL*, pages 263–270.

A. L. Blum and R. L. Rivest. 1992. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127.

A. Culotta and A. McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–751.

S. Dumais. 1998. Using SVMs for text categorization. *IEEE Intelligent Systems Magazine*, 13(4), July/August.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conf. on Machine Learning*, pages 137–142.

P. Kuusela and D. Ocone. 2004. Learning with side information: PAC learning bounds. *J. of Computer and System Sciences*, 68(3):521–545, May.

D. D. Lewis and W. A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proc. of ACM-SIGIR*, pages 3–12.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of ACL*, pages 271–278.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of EMNLP*, pages 79–86.

H. Raghavan, O. Madani, and R. Jones. 2005. Interactive feature selection. In *Proc. of IJCAI*, pages 41–46.

[11]One cannot simply flip those highlighted pixels to white or black, since that would cause new features to fire. Possibly one could simply suppress any feature that depends in any way on the highlighted pixels, but this would take away too many important features, including global features.

Combining Reinforcement Learning with Information-State Update Rules[*]

Peter A. Heeman
Center for Spoken Language Understanding
Oregon Health & Science University
Beaverton OR, 97006, USA
heeman@cslu.ogi.edu

Abstract

Reinforcement learning gives a way to learn under what circumstances to perform which actions. However, this approach lacks a formal framework for specifying hand-crafted restrictions, for specifying the effects of the system actions, or for specifying the user simulation. The information state approach, in contrast, allows system and user behavior to be specified as update rules, with preconditions and effects. This approach can be used to specify complex dialogue behavior in a systematic way. We propose combining these two approaches, thus allowing a formal specification of the dialogue behavior, and allowing hand-crafted preconditions, with remaining ones determined via reinforcement learning so as to minimize dialogue cost.

1 Introduction

Two different approaches have become popular for building spoken dialogue systems. The first is the symbolic reasoning approach. Speech actions are defined in a formal logic, in terms of the situations in which they can be applied, and what effect they will have on the speaker's and the listener's mental state (Cohen and Perrault, 1979; Allen and Perrault, 1980). One of these approaches is the *information state* (IS) approach (Larsson and Traum, 2000). The knowledge of the agent is formalized as the state. The IS state is updated by way of *update rules*, which have *preconditions* and *effects*. The preconditions specify what must be true of the state in order to apply the rule. The effects specify how the state changes as a result of applying the rule. At a minimum, two sets of update rules are used: one set, *understanding rules*, specify the effect of an utterance on the agent's state and a second set, *action* rules, specify which speech action can be performed next. For example, a precondition for asking a question is that the agent does not know the answer to the question. An effect of an answer to a question is that the hearer now knows the answer. One problem with this approach is that although necessary preconditions for speech actions are easy to code, there are typically many speech actions that can be applied at any point in a dialogue. Determining which one is the optimal one is a daunting task for the dialogue designer.

The second approach for building spoken dialogue systems is to use *reinforcement learning* (RL) to automatically determine what action to perform in each different dialogue state so as to minimize some cost function (e.g. Walker, 2000; Levin et al., 2000). The problem with this approach, however, is that it lacks the framework of IS to specify the manner in which the internal state is updated. Furthermore, sometimes no preconditions are even specified for the actions, even though they are obvious to the dialogue designer. Thus RL needs to search over a much larger search space, even over dialogue strategies that do not make any sense. This not only substantially slows down the learning procedure, but also increases the chance of being caught in a locally optimal solution, rather than the global optimal. Furthermore, this large search space will limit the complexity of the domains to which RL can be applied.

In this paper, we propose combining IS and RL. IS update rules are formulated for both the system and the simulated user, thus allowing RL to use a rich formalism for specifying complex dialogue processing. The preconditions on the action rules of the system, however, only need to specify the neces-

[*]The author wishes to thank Fan Yang and Michael English for helpful conversations. Funding from the National Science Foundation under grant IIS-0326496 is gratefully acknowledged.

sary preconditions that are obvious to the dialogue designer. Thus, preconditions on the system's actions might not uniquely identify a single action that should be performed in a given state. Instead, RL is used to determine which of the applicable actions minimizes a dialogue cost function.

In the rest of the paper, we first present an example domain. Section 3 gives an overview of applying RL to dialogue strategy and Section 4 gives an overview of IS. Section 5 demonstrates that IS can be used for simulating a dialogue between the system and a user. Section 6 demonstrates how IS can be used with RL. Section 7 gives results on using hand-crafted preconditions specified in the IS update rules to simplify learning dialogue strategies with RL. Section 8 gives concluding comments.

2 Flight Information Application

To illustrate our proposed approach, we use the flight information domain, similar to that of Levin et al. (2000). The goal of the system is to display a short list of flights that meets the user's requirements. The user is assumed to have a flight in mind, in terms of its destination, origin, airline, departure time, and number of stops. The user might be flexible on some of the parameters. It is assumed that the user will not change his or her mind depending on what flights are found.

In this paper, we are focusing on dialogue management issues, and so we use a semantic representation for both the input and output of the system. The system can ask the user the value of parameter p with 'askconstraint p', and the user will answer with 'constrain p v', where v is the user's preferred value of the parameter.[1] The system can ask whether the user is flexible on the values for parameter p with 'askrelax p', and the user will answer with 'relax p a', where a is either 'yes' or 'no'. The system can do a database query, 'dbquery', to determine whether any flights match the current parameters. If no flights exactly match, 'dbquery' will check if any flights match according to the relaxed restrictions, by ignoring parameters that the system knows the user is flexible on. The system can display the found flights with 'output'. It can also quit at any time. A sample dialogue is given in Fig. 1.

```
system   askconstraint from
user     constrain from miami
system   askconstraint to
user     constrain to sacramento
system   askconstraint departure
user     constrain departure 6pm
system   dbquery miami sacremento - 6pm
system   askconstraint airline
user     constrain airline united
system   dbquery miami sacremento united ...
system   askrelax departure
user     relax departure yes
system   dbquery miami sacremento united ...
system   output {918 11671 13288}
system   finish
```

Figure 1: Sample dialogue

3 Reinforcement Learning (RL)

Given a set of system actions, a set of states, and a cost function that measures the quality of a dialogue, RL searches for an optimal dialogue policy (Sutton and Barto, 1998; Levin et al., 2000).

Cost Function: The cost function assesses how good a dialogue is: the lower the cost, the better the dialogue. RL uses the cost function to provide feedback in its search for an optimal strategy. The cost function is specified by the dialogue designer, and can take into account any number of factors, typically including dialogue length and solution quality.

System Actions: RL takes as input a finite number of actions, and for each state, learns which action is best to perform. The dialogue designer decides what the actions will be, both in terms of how much to combine into a single action, and how specific each action should be.

State Variables: RL learns what system action to perform in each state. The RL states are defined in terms of a set of state variables: different values for the variables define the different states that can exist. The state variables need to include all information that the dialogue designer thinks will be relevant in determining what action to perform next. Any information that is thought to be irrelevant is excluded in order to keep the search space small.

Transitions: RL treats a dialogue as a succession of states, with actions causing a transition from one state to the next. The transition thus encompasses the effect of the system making the speech act,

[1] In contrast to Levin, over-answering by the user is not allowed. The system also does not have a general greeting, to which the user can answer with any of the flight parameters.

the user's response to the system's speech act, and the system's understanding of the user's response. Hence, the transition incorporates a *user simulation*. In applying RL to dialogue policies, the transition from a state-action pair to the next state is usually modeled as a probability distribution, and is not further decomposed (e.g. Levin et al., 2000).

Policy Exploration: RL searches the space of polices by determining Q for each state-action pair *s-a*, which is the minimal cost to get to the final state from state s starting with action a. From the Q values, a policy can be determined: for each state s, choose the action a that has the maximum Q value.

Q is determined in an iterative fashion. The current estimates for Q for each state-action are used to determine the current dialogue policy. The policy, in conjunction with the transition probabilities, are used to produce a dialogue run, which is a sequence of state-action pairs, each pair having an associated cost to get to the next state-action pair. Thus, for a dialogue run, the cost from each state-action pair to the final state can be determined. These costs are used to revise the Q estimates.

To produce a dialogue run, the ϵ-greedy method is often used. In this approach, with probability ϵ, an action other than the action specified by the current policy is chosen. This helps ensure that new estimates are obtained for all state-action pairs, not just ones in the current policy. Typically, a number of dialogue runs, an *epoch*, are made before the Q values and dialogue policy are updated. With each successive epoch, a better dialogue policy is used, and thus the Q estimates will approach their true values, which in turn, ensures that the dialogue policy is approaching the optimal one.

3.1 Flight Information Task in RL

To illustrate how RL learns a dialogue policy, we use the flight information task from Section 2.

Actions: The system actions were given in Section 2. The queries for the destination, origin, airline, departure time, number of stops are each viewed as different actions so that RL can reason about the individual parameters. There are also 5 separate queries for checking whether each parameter can be relaxed. There is also a database query to determine which flights match the current parameters. This is included as an RL action, even though it is not to the user, so that RL can decide when it should be performed. There is also an output and a finish action.

State Variables: We use the following variables for the RL state. The variable 'fromP' indicates whether the origin has been given by the user and the variable 'fromR' indicates whether the user has been asked if the origin can be relaxed, and if so, what the answer is. Similar variables are used for the other parameters. The variable 'dbqueried' indicates whether the database has been queried. The variable 'current' indicates whether no new parameters have been given or relaxed since the last database query. The variable 'NData' indicates the number of items that were last returned from the database quantized into 5 groups: none, 1-5, 6-12, 13-30, more than 30). The variable 'outputP' indicates whether any flights have been given to the user. Note that the actual values of the parameters are not included in the state. This helps limit the size of the search space, but precludes the values of the parameters from being used in deciding what action to perform next.

Cost Function: Our cost function is the sum of four components. Each speech action has a cost of 1. A database query has a cost of 2 plus 0.01 for each flight found. Displaying flights to the user costs 0 for 5 or fewer flights, 8 for 12 or fewer flights, 16 for 30 or fewer flights, and 25 for 30 or more flights. The last cost is the solution cost. This cost takes into account whether the user's preferred flight is even in the database, and if so, whether it was shown to the user. The solution cost is zero if appropriate information is given to the user, and 90 points otherwise.

3.2 Related Work in RL

In the work of Levin, Pieraccini, and Eckert (2000), RL was used to choose between all actions. Actions that resulted in infelicitous speech act sequences were allowed, such as asking the value of a parameter that is already known, asking if a parameter can be relaxed when the value of the parameter is not even known, or displaying values when a database query has not yet been performed.

In other work, RL has been used to choose among a subset of the actions in certain states (Walker, 2000; Singh et al., 2002; Scheffler and Young, 2002; English and Heeman, 2005). However, no formal framework is given to specify which actions to choose from.

Furthermore, none of the approaches used a formal specification for updating the RL variables after a speech action, nor for expressing the user simulation. As RL is applied to more complex tasks, with more complex speech actions, this will lead to difficulty in encoding the correct behavior.

Georgila, Henderson, and Lemon (2005) advocated the use of IS to specify the dialogue context for learning user simulations needed in RL. However, they did not combine hand-crafted with learned preconditions, and it is unclear whether they used IS to update the dialogue context,

4 Information State (IS)

IS has been concerned with capturing how to update the state of a dialogue system in order to build advanced dialogue systems (Larsson and Traum, 2000). For example, it has been used to build systems that allow for both system and user initiative, over answering, confirmations, and grounding (e.g. (Bohlin et al., 1999; Matheson et al., 2000)). It uses a set of state variables, whose values are manipulated by update rules, run by a control strategy.

State Variables: The state variables specify the knowledge of the system at any point in the dialogue. This is similar to the RL variables, except that they must contain everything that is needed to completely specify the action that the system should perform, rather than just enough information to choose between competing actions. A number of standard variables are typically used to interface to other modules in the system. The variable ‘lastMove’ has the semantic representation of what was last said, either by the user or the system and ‘lastSpeaker’ indicates who spoke the last utterance. Both are read-only. The variable ‘nextMove’ is set by the action rules to the semantic representation of the next move and ‘keepTurn’ is set to indicate whether the current speaker will keep the turn to make another utterance.

Update Rules: Update rules have preconditions and effects. The preconditions specify what must be true of the state in order to apply the rule. The effects specify how the state should be updated. In this paper, we will use two types of rules. Understanding rules will be used to update the state to take into account what was just said, by both the user and the system. Action rules determine what the system will say next and whether it will keep the turn.

Control Strategy: The control strategy specifies how the update rules should be processed. In our example, the control strategy specifies that the understanding rules are processed first, and then the action rules if the system has the turn. The control strategy also specifies which rules should be applied: (a) just the first applicable rule, (b) all applicable rules, or (c) randomly choose one of the applicable rules.

Although there is a toolkit available for building IS systems (Larsson and Traum, 2000), we built a simple version in Tcl. Update rules are written using Tcl code, which allows for simple interpretation of the rules. The state is saved as Tcl variables, and thus allows strings, numbers, booleans, and lists.

4.1 Flight Information Example in IS

We now express the flight information system with the IS approach. This allows for a precise formalization of the actions, both the conditions under which they should be performed and their effects.

The IS state variables are similar to the RL ones given in Section 3. Instead of the variable ‘fromP’, it includes the variable ‘from’, which has the actual value of the parameter if known, and ‘’ otherwise. The same is true for the destination, airline, departure time, and number of stops. Instead of the RL variable ‘NData’ and ‘outputP, ‘results’ holds the actual database and ‘output’ holds the actual flights displayed to the user.

Figure 2 displays the system’s understanding rules, which are used to update the state variables after an utterance is said. Although it is common practice in IS to use understanding rules even for one’s own utterances, the example application is simple enough to do without this. Understanding rules are thus only used for understanding the user’s utterances: giving a parameter value or specifying whether a parameter can be relaxed. As can be seen, any time the user specifies a new parameter or relaxes a parameter, ‘current’ is set to false.

Figure 3 gives the action rules for the system. Rules for querying the destination, departure, and number of stops are not shown; neither are the rules for querying whether the destination, origin, airline, and number of stops can be relaxed. The effects of the rules show how the state is updated if the rule is applied. For most of the rules, this is simply to set ‘nextMove’ and ‘keepTurn’ appropriately. The ‘dbquery’ action is more complicated: it runs the database query and updates ‘results’. It then updates the variables ‘queriedDB’, and ‘current’ appropriately. Note that the actions ‘dbquery’ and ‘output’ specify that the system wants to keep the turn.

The preconditions of the update rules specify the exact conditions under which the rule can be applied. The preconditions on the understanding rules are straightforward, and simply check the user’s response. The preconditions on the action rules are more complex. We divide the preconditions into the 4 groups given below, both to simplify the discussion of the preconditions, and because we use these groupings in Section 7.

Speech Acts: Some of the preconditions capture the conditions under which the action can be performed felicitously (Cohen and Perrault, 1979; Allen and Perrault, 1980). Only ask the value of a parameter if you do not know its value. Only ask if a parameter can be relaxed if you know the value of the parameter. Only output the data if it is still current and more than one flight was found. These preconditions are labeled as ‘sa’ in Fig. 3.

Application Restrictions: These preconditions enforce the specification of the application. For our application, the system should only output data once: once data is output, the system should end the conversation. These preconditions are labeled as ‘app’ in Fig. 3.

Partial Strategy: These preconditions add addition constraints that seem reasonable: ask the ‘to’, ‘from’, and ‘departure’ parameters first; never relax the ‘to’ and ‘from’; and only ask whether ‘airline’ and ‘stops’ can be relaxed if the database has been queried. Furthermore, the system may only output data if (a) the number of flights is between 1 and 5, or (b) the number of flights is greater than 5 and ‘airline’ and ‘stops’ have both been asked. These preconditions are labeled as ‘ps’ in Fig. 3.

Baseline: The last group of preconditions (together with the previous preconditions) uniquely identify a single action to perform in each state, and

Understand Answer to Constrain Question	
Pre:	[lindex $lastMove 0] == “constrain”
Eff:	set [lindex LastMove 1] [lindex LastMove 2]
	set current 0
Understand Yes Answer to Relax	
Pre:	[lindex lastMove 0] == “relax”
	[lindex lastMove 2] == “yes”
Eff:	set [lindex lastMove 1]R yes
	set current 0
Understand No Answer to Relax	
Pre:	[lindex lastMove 0] == “relax”
	[lindex lastMove 2] == “no”
Eff:	set [lindex lastMove 1]R no

Figure 2: Understanding Rules for System

Ask Origin of Flight		
Pre:	$from == ‘’	sa
	$output == ‘’	app
Eff:	set nextMove “askconstraint from”	
	set keepTurn false	
Ask Airline of Flight		
Pre:	$airline == ‘’	sa
	$output == ‘’	app
	$departure != ‘’	ps
	$queriedDB == true	base
	$current == true	base
	[llength $results] $>$ 5	base
Eff:	set nextMove “askconstraint to”	
	set keepTurn false	
Ask Whether Departure Time can be Relaxed		
Pre:	$departure != ‘’	sa
	$departureR == ‘’	sa
	$output != ‘’	app
	$queriedDB == true	base
	$current == true	base
	$results == {}	base
Eff:	set nextMove ‘askrelax from’	
	set keepTurn false	
Query the Database		
Pre:	$current == false	sa
	$output == ‘’	app
	$departure != ‘’	ps
Eff:	set results [DBQuery $from $to $airline ...]	
	set queriedDB true	
	set current true	
	set nextMove dbquery	
	set keepTurn true	
Output Results to User		
Pre:	$current == true	sa
	$results != {}	sa
	$output == ‘’	app
	[llength $results] $<$ 6 || ([llength $results] $>$ 5 && $airline != “” && $stops != “”)	ps
Eff:	set nextMove “output $results”	
	set output $results	
Finish		
Pre:	$output != ‘”	app
Eff:	set nextMove finish	
Quit		
Pre:	$output == ‘’	app
	$current == true	app
	$results == {}	app
	$airline != ‘’ || $airlineR != ‘’	base
	$stops != ‘’ || $stopsR != ‘’	base
Eff:	set nextMove finish	

Figure 3: Action Rules for System

thus completely specifies a strategy. These are labeled as ‘base’ in Fig. 3. The strategy that we give is based on the optimal strategy found by Levin et al. (2000). After the system asks the values for the ‘from’, ‘to’, and ‘departure’ variables, it then performs a database query. If there are between 1 and 5 flights found, they are displayed to the user. If there are more than 5, the system asks the value of ‘airline’ if unknown, otherwise, ‘number of stops’. If there are 0 items, it tries to relax one of ‘departure’, ‘airline’, and ‘stops’, in that order (but not ‘from’ or ‘to’). Any time new information is gained, such as a parameter value or a parameter is relaxed, the database is requeried, and the process repeats.

5 Implementing the Simulated User

Normally, with IS, the system is run against an actual user, and so no state variables nor update rules are coded for the user. To allow the combination of IS with RL, we need to produce dialogues between the system and a simulated user. As the IS approach is very general, we will use it for implementing the simulated user as well. In this way, we can code the user simulation with a well-defined formalism, thus allowing complex user behaviors. Hence, two separate IS instantiations will be used: one for the system and one for the user. The system’s rules will update the system’s state variables, and the user’s rules will update the user’s state variables; but the two instantiations will be in lock-step with each other.

We built a simulator that runs the system’s rules against the user’s. The simulator (a) runs the understanding rules for the system and the user on the last utterance; then (b) checks who has the turn, and runs that agent’s action rules; and then (c) updates ‘lastSpeaker’ and ‘lastMove’. It repeats these three steps until the ‘finish’ speech act is seen.

5.1 Flight Information Task

The user has the variables ‘from’, ‘to’, ‘departure’, ‘airline’, and ‘stops’, which hold the user’s ideal flight, and are set before the dialogue begins. The variables ‘fromR’, ‘toR’, ‘departureR’, ‘airlineR’, and ‘stopsR’ are also used, and are also set before the dialogue begins. No other variables are used.

For the flight application, separate update rules are used for the user. There are two types of queries

```
Answer Constrain Question
Pre:  [lindex $lastMove 0] == “askconstraint”
Eff:  set nextMove “constraint [lindex $lastmove 1]
              [set [lindex $lastmove 1]]”
      set haveTurn 0
Answer Relax Question
Pre:  [lindex $lastMove 0] == “askrelax”
Eff:  set nextMove “relax [lindex $lastmove 1]
              [set [lindex $lastMove 1]R]”
      set haveTurn 0
```

Figure 4: Action Rules for User

to which the user needs to react, namely, ‘askcontraint’ and ‘askrelax’. This domain is simple enough that we do not need separate understanding and action rules, and so we encompass all reasoning in the action rules, shown in Fig. 4. The first rule is for answering system queries about the value of a parameter. The second is for answering queries about whether a parameter can be relaxed.

6 Combining IS and RL

RL gives a way to learn the best action to perform in any given state. However, RL lacks a formal framework for specifying (a) the effects of the system’s actions, (b) hand-crafted preconditions of the system’s actions, and (c) the simulated user. Hence, we combine RL and IS to rectify these deficits. IS update rules are formulated for both the system and the simulated user, as done in Section 5.1. The preconditions on the system’s action rules, however, only need to specify a subset of the preconditions, ones that are obvious the dialogue designer. The rest of the preconditions will be determined by RL, so as to minimize a cost function. To combine these two approaches, we need to (a) resolve how the IS and RL state transitions relate to each other; (b) resolve how the IS state relates to the RL state; and (c) specify how utterance costs can be specified in the general framework of IS.

Transitions: When using IS for both the system and user simulation, the state transitions for each are happening in lock-step (Section 5.1). In combining RL and IS, the RL transitions happen at a courser granularity than the IS transitions, and group together everything that happens between two successive system actions. Thus, the RL states are those IS states just before a system action.

State Variables: For the system, we add all of the RL variables to the IS variables, and remove any duplicates. The RL variables are thus a subset of the IS variables. Some of the variables might be simplifications of other variables. For our flight example, we have the exact values of the origin, destination, airline, departure time, and number of stops, as well as a simplification of each that only indicates whether the parameter has been given or not.

Rather than have the system's IS rules update all of the variables, we allow variables to be declared as either *primitive* or *derived*.[2] Only primitive variables are updated by the effects of the update rules. The derived variables are re-computed from the primitive ones each time an update rule is applied. For our flight example, the variables 'fromP', 'toP', 'airlineP', 'departureP', 'stopsP', 'outputP', and 'NData' are derived variables, and these are updated via a procedure.

As the RL variables are a subset of the IS variables, the RL states are coarser than the IS states. We do not allow hand-crafted preconditions in the system's action rules to distinguish at the finer granularity. If they did, we would have an action that is only applicable in part of an RL state, and not the rest of it. However, RL needs to find a single action that will work for the entire RL state, and so that action should not be considered. To prevent such problems, the hand-crafted preconditions can only test the values of the RL variables, and not the full set of IS variables. Hence, we rewrote the preconditions in the action rules of Fig. 3 to use the RL variables. This restriction does not apply to the system's understanding rules, nor to the user rules, as those are not subject to RL.

Cost Function: RL needs to track the costs incurred in the dialogue. Rather than leaving this to be specified in an ad-hoc way, we include state variables to track the components of the cost. This way, each update rule can set them to reflect the cost of the rule. Just as with other interface variables (e.g. 'keepTurn'), these are write-only. For our flight example, the output action computes the cost of displaying flights to the user, and the database query action computes the cost of doing the database lookup.

[2]This same distinction is sometimes used in the planning literature (Poole et al., 1998).

7 Evaluation

To show the usefulness of starting RL with some of the preconditions hand-crafted, we applied RL using four different sets of action schemes. The first set, 'none', includes no preconditions on any of the system's actions. The second through fourth sets correspond to the precondition distinctions in Fig. 3, of 'speech act', 'application' and 'partial strategy'.

For each set of action schemas, we trained 30 dialogue policies using an epoch size of 100. Each dialogue was run with the ϵ-greedy method, with ϵ set at 0.15. After certain epochs, we ran the learned policy 2500 times strictly according to the policy. We found that policies did not always converge. Hence, we trained the policies for each set of preconditions for enough epochs so that the average cost no longer improved. More work is needed to investigate this issue.

The results of the simulations are given in Table 1. The first row reports the average dialogue cost that the 30 learned policies achieved. We see that all four conditions achieved an average cost less than the baseline strategy of Fig. 3, which was 17.17. The best result was achieved by the 'application' preconditions. This is probably because 'partial' included some constraints that were not optimal, while the search strategy was not adequate to deal with the large search space in 'speech acts' and 'none'.

The more important result is in the second row of Table 1. The more constrained precondition sets result in significantly fewer states being explored, ranging from 275 for the 'partial' preconditions, up to 18,206 for no preconditions. In terms of number of potential policies explored (computed as the product of the number of actions explored in each state), this ranges from 10^{58} to 10^{7931}. As can be seen, by placing restrictions on the system actions, the space that needs to be explored is substantially reduced.

The restriction in the size of the search space affects how quickly RL takes to find a good solution. Figure 5 shows how the average cost for each set of

	None	SA	App.	Partial
Dialogue Cost	16.65	16.95	15.24	15.68
States Explored	18206	5261	4080	275
Policies ($\log_{10}$)	7931	2008	1380	58.7

Table 1: Comparison of Preconditions

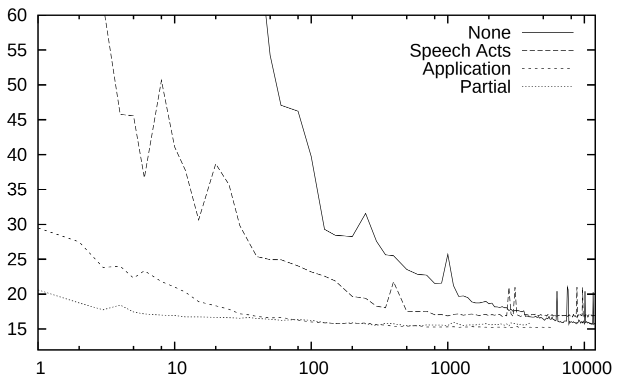


Figure 5: Average dialogue cost versus epochs

preconditions improved with the number of epochs. As can be seen, by including more preconditions in the action definitions, RL is able to find a good solution more quickly. For the 'partial' preconditions, after 10 epochs, RL achieves a cost less than 17.0. For the 'application' setting, this does not happen until 40 epochs. For 'speech act', it takes 1000 epochs, and for 'none', it takes 3700 epochs. So, adding hand-crafted preconditions allows RL to converge more quickly.

8 Conclusion

In this paper, we demonstrated how RL and IS can be combined. From the RL standpoint, this allows the rich formalism of IS update rules to be used for formalizing the effects of the system's speech actions, and for formalizing the user simulation, thus enabling RL to be applied to domains that require complex dialogue processing. Second, use of IS allows obvious preconditions to be easily formulated, thus allowing RL to search a much smaller space of policies, which enables it to converge more quickly to the optimal policy. This should also enable RL to be applied to complex domains with large numbers of states and actions.

From the standpoint of IS, use of RL means that not all preconditions need be hand-crafted. Preconditions that capture how one action might be more beneficial than another can be difficult to determine for dialogue designers. For example, knowing whether to first ask the number of stops or the airline, depends on the characteristics of the flights in the database, and on users' relative flexibility with these two parameters. The same problems occur for knowing under which situations to requery the database or ask for another parameter. RL solves this issue as it can explore the space of different policies to arrive at one that minimizes a dialogue cost function.

References

J. Allen and C. Perrault. 1980. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178.

P. Bohlin, R. Cooper, E. Engdahl, and S. Larsson. 1999. Information states and dialogue move engines. In *Proceedings of the IJCAI Workshop: Knowledge and Reasoning in Practical Dialogue Systems*, pg. 25–31.

P. Cohen and C. Perrault. 1979. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212.

M. English and P. Heeman. 2005. Learning mixed initiative dialog strategies by using reinforcement learning on both conversants. In *HLT and EMNLP*, pages 1011–1018, Vancouver Canada, October.

K. Georgila, J. Henderson, and O. Lemon. 2005. Learning user simulations for information state update dialogue systems. In *Eurospeech*, Lisbon Portugal.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6:323–340.

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.

C. Matheson, M. Poesio, and D. Traum. 2000. Modelling grounding and discourse obligations using update rules. In *NAACL*, Seattle, May.

D. Poole, A. Mackworth, and R. Goebel. 1998. *Computational Intelligence: a logical approach*. Oxford University Press.

K. Scheffler and S. J. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *HLT*, pg. 12–18, San Diego.

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.

M. Walker. 2000. An application of reinforcement learning to dialog strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416.

Estimating the Reliability of MDP Policies: A Confidence Interval Approach

Joel R. Tetreault
University of Pittsburgh
LRDC
Pittsburgh PA, 15260, USA
tetreaul@pitt.edu

Dan Bohus
Carnegie Mellon University
Dept. of Computer Science
Pittsburgh, PA, 15213, USA
dbohus@cs.cmu.edu

Diane J. Litman
University of Pittsburgh
Dept. of Computer Science
LRDC
Pittsburgh PA, 15260, USA
litman@cs.pitt.edu

Abstract

Past approaches for using reinforcement learning to derive dialog control policies have assumed that there was enough collected data to derive a reliable policy. In this paper we present a methodology for numerically constructing confidence intervals for the expected cumulative reward for a learned policy. These intervals are used to (1) better assess the reliability of the expected cumulative reward, and (2) perform a refined comparison between policies derived from different Markov Decision Processes (MDP) models. We applied this methodology to a prior experiment where the goal was to select the best features to include in the MDP state-space. Our results show that while some of the policies developed in the prior work exhibited very large confidence intervals, the policy developed from the best feature set had a much smaller confidence interval and thus showed very high reliability.

1 Introduction

NLP researchers frequently have to deal with issues of data sparsity. Whether the task is machine translation or named-entity recognition, the amount of data one has to train or test with can greatly impact the reliability and robustness of one's models, results and conclusions.

One research area that is particularly sensitive to the data sparsity issue is machine learning, specifically in using Reinforcement Learning (RL) to learn the optimal action for a dialogue system to make given any user state. Typically this involves learning from previously collected data or interacting in real-time with real users or user simulators. One of the biggest advantages to this machine learning approach is that it can be used to generate optimal policies for every possible state. However, this method requires a thorough exploration of the state-space to make reliable conclusions on what the best actions are. States that are infrequently visited in the training set could be assigned sub-optimal actions, and therefore the resulting dialogue manager may not provide the best interaction for the user.

In this work, we present an approach for estimating the reliability of a policy derived from collected training data. The key idea is to take into account the uncertainty in the model parameters (MDP transition probabilities), and use that information to numerically construct a confidence interval for the expected cumulative reward for the learned policy. This confidence interval approach allows us to: (1) better assess the reliability of the expected cumulative reward for a given policy, and (2) perform a refined comparison between policies derived from different MDP models.

We apply the proposed approach to our previous work (Tetreault and Litman, 2006) in using RL to improve a spoken dialogue tutoring system. In that work, a dataset of 100 dialogues was used to develop a methodology for selecting which user state features should be included in the MDP state-space. But are 100 dialogues enough to generate reliable policies? In this paper we apply our confidence in-

Proceedings of NAACL HLT 2007, pages 276–283,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

terval approach to the same dataset in an effort to investigate how reliable our previous conclusions are, given the amount of available training data.

In the following section, we discuss the prior work and its data sparsity issue. In section 3, we describe in detail our confidence interval methodology. In section 4, we show how this methodology works by applying it to the prior work. In sections 5 and 6, we present our conclusions and future work.

2 Previous Work

Past research into using RL to improve spoken dialogue systems has commonly used Markov Decision Processes (MDP's) (Sutton and Barto, 1998) to model a dialogue (such as (Levin and Pieraccini, 1997) and (Singh et al., 1999)).

A MDP is defined by a set of states $\{s_i\}_{i=1..n}$, a set of actions $\{a_k\}_{k=1..p}$, and a set of transition probabilities which reflect the dynamics of the environment $\{p(s_i|s_j, a_k)\}_{i,j=1..n}^{k=1..p}$: if the model is at time t in state s_j and takes action a_k, then it will transition to state s_i with probability $p(s_i|s_j, a_k)$. Additionally, an expected reward $r(s_i, s_j, a_k)$ is defined for each transition. Once these model parameters are known, a simple dynamic programming approach can be used to learn the optimal control policy π^*, i.e. the set of actions the model should take at each state, to maximize its expected cumulative reward.

The dialog control problem can be naturally cast in this formalism: the states $\{s_i\}_{i=1..n}$ in the MDP correspond to the dialog states (or an abstraction thereof), the actions $\{a_k\}_{k=1..p}$ correspond to the particular actions the dialog manager might take, and the rewards $r(s_i, s_j, a_k)$ are defined to reflect a particular dialog performance metric. Once the MDP structure has been defined, the model parameters $\{p(s_i|s_j, a_k)\}_{i,j=1..n}^{k=1..p}$ are estimated from a corpus of dialogs (either real or simulated), and, based on them, the policy which maximizes the expected cumulative reward is computed.

While most work in this area has focused on developing the best policy (such as (Walker, 2000), (Henderson et al., 2005)), there has been relatively little work done with respect to selecting the best features to include in the MDP state-space. For instance, Singh et al. (1999) showed that dialogue length was a useful state feature and Frampton and Lemon (2005) showed that the user's last dialogue act was also useful. In our previous work, we compare the worth of several features. In addition, Paek and Chickering's (2005) work showed how a state-space can be reduced by only selecting features that are relevant to maximizing the reward function.

The motivation for this line of research is that if one can properly select the most informative features, one develops better policies, and thus a better dialogue system. In the following sections we summarize our past data, approach, results, and issue with policy reliability.

2.1 MDP Structure

For this study, we used an annotated corpus of human-computer spoken dialogue tutoring sessions. The fixed-policy corpus contains data collected from 20 students interacting with the system for five problems (for a total of 100 dialogues of roughly 50 turns each). The corpus was annotated with 5 state features (Table 1). It should be noted that two of the features, Certainty and Frustration, were manually annotated while the other three were done automatically. All features are binary except for Certainty which has three values.

State	Values
Correctness	Student is correct or incorrect in the current turn
Certainty	Student is certain, neutral or uncertain in the current turn
Concept Repetition	A particular concept is either new or repeated
Frustration	Student is frustrated or not in the current turn
Percent Correct	Student answers over 66% of questions correctly in dialogue so far, or less

Table 1: State Features in Tutoring Corpus

For the action set $\{a_k\}_{k=1..p}$, we looked at what type of question the system could ask the student given the previous state. There are a total of four possible actions: ask a short answer question (one that requires a simple one word response), a complex answer question (one that requires a longer, deeper response), ask both a simple and complex question in the same turn, or do not ask a question at all (give a hint). The reward function r was the learning gain of each student based on a pair of tests before and after the entire session of 5 dialogues. The 20 students were split into two groups (high and low learners) based on their learning gain, so 10 students and their respective five dialogues were given a positive reward of +100, while the remainder were assigned a negative reward of -100. The rewards were assigned in the final dialogue state, a common approach when applying RL in spoken dialogue systems.

2.2 Approach and Results

To investigate the usefulness of different features, we took the following approach. We started with two baseline MDPs. The first model (Baseline 1) used only the Correctness feature in the state-space. The second model (Baseline 2) included both the Correctness and Certainty features. Next we constructed 3 new models by adding each of the remaining three features (Frustration, Percent Correct and Concept Repetition) to the Baseline 2 model.

We defined three metrics to compare the policies derived from these MDPs: (1) Diff's: the number of states whose policy differs from the Baseline 2 policy, (2) Percent Policy change (P.C.): the weighted amount of change between the two policies (100% indicates total change), and (3) Expected Cumulative Reward (or ECR) which is the average reward one would expect in that MDP when in the state-space.

The intuition is that if a new feature were relevant, the corresponding model would lead to a different policy and a better expected cumulative reward (when compared to the baseline models). Conversely, if the features were not useful, one would expect that the new policies would look similar (specifically, the Diff's count and % Policy Change would be low) or produce similar expected cumulative rewards to the original baseline policy.

The results of this analysis are shown in Table 2 [1] The Diff's and Policy Change metrics are undefined for the two baselines since we only use these two metrics to compare the other three features to Baseline 2. All three metrics show that the best feature to add to the Baseline 2 model is Concept Repetition since it results in the most change over the Baseline 2 policy, and also the expected reward is the highest as well. For the remainder of this paper, when we refer to Concept Repetition, Frustration, or Percent Correctness, we are referring to the model that includes that feature as well as the Baseline 2 features Correctness and Certainty.

State Feature	# Diff's	% P.C.	ECR
Baseline 1	N/A	N/A	6.15
Baseline 2	N/A	N/A	31.92
B2 + Concept Repetition	10	80.2%	42.56
B2 + Frustration	8	66.4%	32.99
B2 + Percent Correctness	4	44.3%	28.50

Table 2: Feature Comparison Results

2.3 Problem with Reliability

However, the approach discussed above assumes that given the size of the data set, the ECR and policies are reliable. If the MDP model were very fragile, that is the policy and expected cumulative reward were very sensitive to the quality of the transition probability estimates, then the metrics could reveal quite different rankings. Previously, we used a qualitative approach of tracking how the worth of each state (V-value) changed over time. The V-values indicate how much reward one would expect from starting in that state to get to a final state. We hypothesized that if the V-values stabilized as data increased, then the learned policy would be more reliable.

So is this V-value methodology adequate for assessing if there is enough data to determine a stable policy, and also for assessing if one model is better than another? Since our approach for state-space selection is based on comparing a new policy with a baseline policy, having a stable policy is extremely important since instability could lead to different conclusions. For example, in one comparison, a new policy could differ with the baseline in 8 out of 10 states. But if the MDP were unstable, adding just a little more data could result in a difference of only 4 out of 10 states. Is there an approach that can categorize whether given a certain data size,

[1] Please note that to due to refinements in code, there is a slight difference between the ECR's reported in this work and the ECR's reported in the previous work, for the three features added to Baseline 2. These changes did not alter the rankings of these models, or the conclusions of the previous work.

that the expected cumulative reward (and thus the policy) is reliable? In the next section we present a new methodology for numerically constructing confidence intervals for these value function estimates. Then, in the following section, we reevaluate our prior work with this methodology and discuss the results.

3 Confidence Interval Methodology

3.1 Policy Evaluation with Confidence Intervals

The starting point for the proposed methodology is the observation that for each state s_j and action a_k in the MDP, the set of transition probabilities $\{p(s_i|s_j, a_k)\}_{i=1..n}$ are modeled as multinomial distributions that are estimated from the transition counts in the training data:

$$\hat{p}(s_i|s_j, a_k) = \frac{c(s_i, s_j, a_k)}{\sum_{i=1}^{n} c(s_i, s_j, a_k)} \tag{1}$$

where n is the number of states in the model, and $c(s_i, s_j, a_k)$ is the number of times the system was in state s_j, took action a_k, and transitioned to state s_i in the training data.

It is important to note that these parameters are just estimates. The reliability of these estimates clearly depends on the amount of training data, more specifically on the transition counts $c(s_i, s_j, a_k)$. For instance, consider a model with 3 states and 2 actions. Say the model was in state s_1 and took action a_1 ten times. Out of these, three times the model transitioned back to state s_1, two times it transitioned to state s_2, and five times to state s_3. Then we have:

$$\hat{p}(s_i|s_1, a_1) = \langle 0.3; 0.2; 0.5 \rangle = \langle \frac{3}{10}; \frac{2}{10}; \frac{5}{10} \rangle \tag{2}$$

Additionally, let's say the same model was in state s_2 and took action a_2 1000 times. Following that action, it transitioned 300 times to state s_1, 200 times to state s_2, and 500 times to state s_3.

$$\hat{p}(s_i|s_2, a_2) = \langle 0.3; 0.2; 0.5 \rangle = \langle \frac{300}{1000}; \frac{200}{1000}; \frac{500}{1000} \rangle \tag{3}$$

While both sets of transition parameters have the same value, the second set of estimates is more reliable. The central idea of the proposed approach is to model this uncertainty in the system parameters, and use it to numerically construct confidence intervals for the value of the optimal policy.

Formally, each set of transition probabilities $\{p(s_i|s_j, a_k)\}_{i=1..n}$ is modeled as a multinomial distribution, estimated from data[2]. The uncertainty of multinomial estimates are commonly modeled by means of a Dirichlet distribution. The Dirichlet distribution is characterized by a set of parameters α_1, α_2, ..., α_n, which in this case correspond to the counts $\{c(s_i, s_j, a_k)\}_{i=1..n}$. For any given j, the likelihood of the set of multinomial transition parameters $\{p(s_i|s_j, a_k)\}_{i=1..n}$ is then given by:

$$P(\{p(s_i|s_j, a_k)\}_{i=1..n}|D) = \\ = \frac{1}{Z(D)} \prod_{i=1}^{n} p(s_i|s_j, a_k)^{\alpha_i - 1} \tag{4}$$

where $Z(D) = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)}$ and $\alpha_i = c(s_i, s_j, a_k)$. Note that the maximum likelihood estimates for the formula above correspond to the frequency count formula we have already described:

$$\hat{p}_{ML}(s_i|s_j, a_k) = \frac{\alpha_i}{\sum_{i=1}^{n} \alpha_i} = \frac{c(s_i, s_j, a_k)}{\sum_{i=1}^{n} c(s_i, s_j, a_k)} \tag{5}$$

To capture the uncertainty in the model parameters, we therefore simply need to store the counts of the observed transitions $c(s_i, s_j, a_k)$. Based on this model of uncertainty, we can numerically construct a confidence interval for the value of the optimal policy π^*. Instead of computing the value of the policy based on the maximum likelihood transition estimates $\hat{T}_{ML} = \{\hat{p}_M L(s_i|s_j, a_k)\}_{i,j=1..n}^{k=1..p}$, we generate a large number of transition matrices $\hat{T}_1$, $\hat{T}_1$, ... $\hat{T}_m$ by sampling from the Dirichlet distributions corresponding to the counts observed in the training data (in the experiments reported in this paper, we used $m = 1000$). We then compute the value of the optimal policy π^* in each of these models $\{V_{\pi^*}(\hat{T}_i)\}_{i=1..m}$. Finally, we numerically construct the 95% confidence interval for the value function based on the resulting value estimates: the bounds for the confidence interval are set at the lowest and highest 2.5 percentile of the resulting distribution of the values for the optimal policy $\{V_{\pi^*}(\hat{T}_i)\}_{i=1..m}$.

The algorithm is outlined below:

[2]By p we will denote the true model parameters; by $\hat{p}$ we will denote data-driven estimates for these parameters

1. compute transition counts from the training set:

$$C = \{c(s_i, s_j, a_k)\}_{i,j=1..n}^{k=1..p} \tag{6}$$

2. compute maximum likelihood estimates for transition probability matrix:

$$\hat{T}_{ML} = \{\hat{p}_{ML}(s_i|s_j, a_k)\}_{i,j=1..n}^{k=1..p} \tag{7}$$

3. use dynamic programming to compute the optimal policy π^* for model $\hat{T}_{ML}$

4. sample m transition matrices $\{\hat{T}_k\}_{k=1..m}$, using the Dirichlet distribution for each row:

$$\{\hat{p}_i(s_i|s_j, a_k)\}_{i=1..n} = \\ = Dir(\{c(s_i, s_j, a_k)\}_{i=1..n}) \tag{8}$$

5. evaluate the optimal policy π^* in each of these m models, and obtain $V_{\pi^*}(\hat{T}_i)$

6. numerically build the 95% confidence interval for V_{π^*} from these estimates.

To summarize, the central idea is to take into account the reliability of the transition probability estimates and construct a confidence interval for the expected cumulative reward for the learned policy. In the standard approach, we would compute an estimate for the expected cumulative reward, by simply using the transition probabilities derived from the training set. Note that these transition probabilities are simply estimates which are more or less accurate, depending on how much data is available. The proposed methodology does not fully trust these estimates, and asks the question: given that the real world (i.e. real transition probabilities) might actually be a bit different than we think it is, how well can we expect the learned policy to perform? Note that the confidence interval we construct, and therefore the conclusions we draw, are with respect to the policy learned from the current estimates, i.e. from the current training set. If more data becomes available, a different optimal policy might emerge, about which we cannot say much.

3.2 Related Work

Given the stochastic nature of the models, confidence intervals are often used to estimate the reliability of results in machine learning experiments, e.g. (Rivals and Personnaz, 2002), (Schapire, 2002) and (Dumais et al., 1998). In this work we use a confidence interval methodology in the context of MDPs. The idea of modeling the uncertainty of the transition probability estimates using Dirichlet models also appears in (Jaulmes et al., 2005). In that work, the authors used the uncertainty in model parameters to develop active learning strategies for partially observable MDPs, a topic not previously addressed in the literature. In our work we rely on the same model of uncertainty for the transition matrix, but use it to derive confidence intervals for the expected cumulative reward for the learned optimal policy, in an effort to assess the reliability of this policy.

4 Results

Our previous results indicated that Concept Repetition was the best feature to add to the Baseline 2 state-space model, but also that Percent Correctness and Frustration (when added to Baseline 2) offered an improvement over the Baseline MDP's. However, these conclusions were based on a very qualitative approach for determining if a policy is reliable or not. In the following subsection, we apply our approach of confidence intervals to empirically determine if given this data set of 100 dialogues, whether the estimates of the ECR are reliable, and whether the original rankings and conclusions hold up under this refined analysis. In subsection 4.2, we provide a methodology for pinpointing when one model is better than another.

4.1 Quantitative Analysis of ECR Reliability

For our first investigation, we look at the confidence intervals of each MDP's ECR over the entire data set of 20 students (later in this section we show plots for the confidence intervals as data increases). Table 3 shows the upper and lower bounds for the ECR originally reported in Table 2. The first column shows the original, estimated ECR of the MDP and the last column is the width of the bound (the difference between the upper and lower bound).

So what conclusions can we make about the reliability of the ECR, and hence of the learned policies for the different MDP's, given this amount of training data? The confidence interval for the ECR for

State Feature	ECR	Lower Bound	Upper Bound	Width
Baseline 1	6.15	0.21	23.73	23.52
Baseline 2 (B2)	31.92	-5.31	60.48	65.79
B2 + Concept Repetition	42.56	28.37	59.29	30.92
B2 + Frustration	32.99	-4.12	61.30	65.42
B2 + Percent Correctness	28.50	-5.89	57.82	63.71

Table 3: Confidence Intervals with complete dataset

the Baseline 1 model ranges from 0.21 to 23.73. Recall that the final states are capped at +100 and -100, and are thus the maximum and minimum bounds that one can see in this experiment. These bounds tell us that, if we take into account the uncertainty in the model estimates (given the small training set size), with probability 0.95 the actual true ECR for this policy will be greater than 0.21 and smaller than 23.73. The width of this confidence interval is 23.52.

For the Baseline 2 model, the bounds are much wider: from -5.31 to 60.48, for a total width of 65.79. While the ECR estimate is 31.92 (which is seemingly larger than 6.15 for the Baseline 1 model), the wide confidence interval tells us that this estimate is not very reliable. It is possible that the policy derived from this model with this amount of data could perform poorly, and even get a negative reward. From the dialogue system designer's standpoint, a model like this is best avoided.

Of the remaining three models – Concept Repetition, Frustration, and Percent Correctness, the first one exhibits a tighter confidence interval, indicating that the estimated expected cumulative reward (42.56) is fairly reliable: with 95% probability of being between 28.37 and 59.29. The ECR for the other two models (Frustration and Percent Correctness) again shows a wide confidence interval once we take into account the uncertainty in the model parameters.

These results shed more light on the shortcomings of the ECR metric used to evaluate the models in prior work. This estimate does not take into account the uncertainty of the model parameters. For example, a model can have an optimal policy with a very high ECR value, but have very wide confidence bounds reaching even into negative rewards. On the other hand, another model can have a relatively lower ECR but if its bounds are tighter (and the lower bound is not negative), one can know that that policy is less affected by poor parameter estimates stemming from data sparsity issues. Using the confidence intervals associated with the ECR gives a much more refined, quantitative estimate of the reliability of the reward, and hence of the policy derived from that data.

An extension of this result is that confidence intervals can also allow us to make refined judgments about the comparative utility of different features, the original motivation of our prior study. Basically, a model (M1) is better than another (M2) if M1's lower bound is greater than the upper bound of M2. That is, one knows that 95% of the time, the worst case situation of M1 (the lower bound) will always yield a higher reward than the best case of M2. In our data, this happens only once, with Concept Repetition being empirically better than Baseline 1, since the lower bound of Concept Repetition is 28.37 and the upper bound of Baseline 1 is 23.73. Given this situation, Concept Repetition is a useful feature which, when included in the model, leads to a better policy than simply using Correctness. We cannot draw any conclusions about the other features, since their bounds are generally quite wide. Given this amount of training data, we cannot say whether Percent Correctness and Frustration are better features than the Baseline MDP's. Although their ECR's are higher, there is too much uncertainty to definitely conclude they are better.

4.2 Pinpointing Model Cross-over

The previous analysis focused on a quantitative method of (1) determining the reliability of the MDP ECR estimate and policy, as well as (2) assessing whether one model is better than another. In this section, we present an extension to the second contribution by answering the question: given that one model is more reliable than another, is it possible to determine at which point one model's estimates become more reliable than another model's? In our

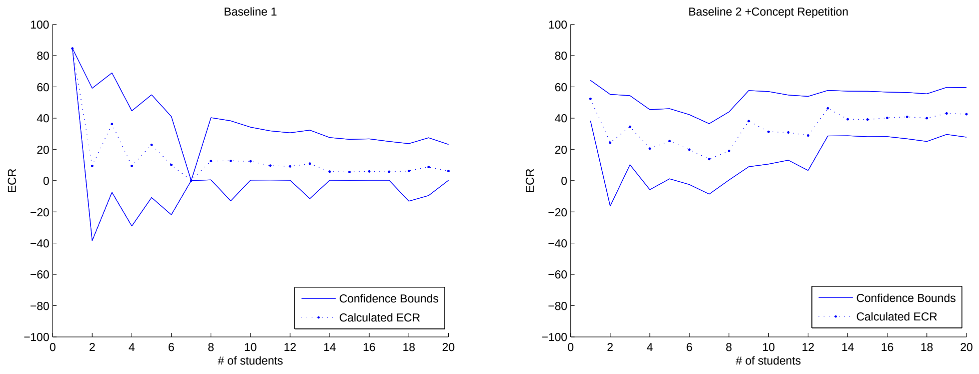


Figure 1: Confidence Interval Plots

case, we want to know at what point Concept Repetition becomes more reliable than Baseline 1. To do this, we investigate how the confidence interval changes as the amount of training data increases instead of looking at the reliability estimate at only one particular data size.

We incrementally increase the amount of training data (adding the data from one new student at a time), and calculate the corresponding optimal policy and confidence interval for the expected cumulative reward for that policy. Figure 1 shows the confidence interval plots as data is added to the MDP for the Baseline 1 and Concept Repetition MDP's. For reference, Baseline 2, Percent Correctness and Frustration plots did not exhibit the same converging behavior as these two, which is not surprising given how wide the final bounds are. For each plot, the bold lines represent the upper and lower bounds, and the dotted line represents the calculated ECR.

Analyzing the two MDP's, we find that the confidence intervals for Baseline 1 and Concept Repetition converge as more data is added, which is an expected trend. One useful result from observing the change in confidence intervals is that one can determine the point in one which one model becomes empirically better than another. Superimposing the upper and lower bounds (Figure 2) reveals that after we include the data from the first 13 students, the lower bound of Concept Repetition crosses over the upper bound of Baseline 1.

Observing this behavior is especially useful for performing model switching. In automatic model switching, a dialogue manager runs in real time and as it collects data, it can switch from using a simple dialogue model to a complex model. Confidence intervals can be used to determine when to switch from one model to the next by checking if a complex model's bounds cross over the bounds of the current model. Basically, the dialogue manager switches when it can be sure that the more complex model's ECR is not only higher, but statistically significantly so.

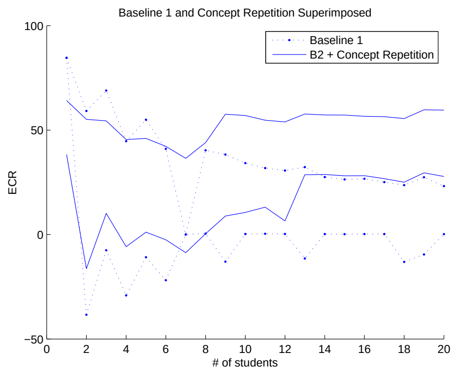


Figure 2: Baseline 1 and Concept Repetition Bounds

5 Conclusions

Past work in using MDP's to improve spoken dialogue systems have usually glossed over the issue of whether or not there was enough training data to develop reliable policies. In this work, we present a numerical method for building confidence intervals for the expected cumulative reward for a learned policy. The proposed approach allows one to (1) better assess the reliability of the expected cumulative reward for a given policy, and (2) perform a refined comparison between policies derived from different MDP models.

We applied this methodology to a prior experiment where the objective was to select the best features to include in the MDP state-space. Our results show that policies constructed from the Baseline 1 and Concept Repetition models are more reliable, given the amount of data available for training. The Concept Repetition model (which is composed of the Concept Repetition, Certainty and Correctness features) was especially useful, as it led to a policy that outperformed the Baseline 1 model, even when we take into account the uncertainty in the model estimates caused by data sparsity. In contrast, for the Baseline 2, Percent Correctness, and Frustration models, the estimates for the expected cumulative reward are much less reliable, and no conclusion can be reliably drawn about the usefulness of these features. In addition, we showed that our confidence interval approach has applications in another MDP problem: model switching.

6 Future Work

As an extension of this work, we are currently investigating in more detail what makes some MDP's reliable or unreliable for a certain data size (such as the case where Baseline 2 does not converge but a more complicated model does, such as Concept Repetition). Our initial findings indicate that, as more data becomes available the bounds tighten for most parameters in the transition matrix. However, for some of the parameters the bounds can remain wide, and that is enough to keep the confidence interval for the expected cumulative reward from converging.

Acknowledgments

We would like to thank Jeff Schneider, Drew Bagnell, Pam Jordan, as well as the ITSPOKE and Pitt NLP groups, and the Dialog on Dialogs group for their help and comments. Finally, we would like to thank the four anonymous reviewers for their comments on the initial version of this paper. Support for this research was provided by NSF grants #0325054 and #0328431.

References

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Conference on Information and Knowledge Management*.

M. Frampton and O. Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

J. Henderson, O. Lemon, and K. Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*.

R. Jaulmes, J. Pineau, and D. Precup. 2005. Active learning in partially observable markov decision processes. In *European Conference on Machine Learning*.

E. Levin and R. Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogues. In *Proc. of EUROSPEECH '97*.

T. Paek and D. Chickering. 2005. The markov assumption in spoken dialogue management. In *6th SIGDial Workshop on Discourse and Dialogue*.

I. Rivals and L. Personnaz. 2002. Construction of confidence intervals for neural networks based on least squares estimation. In *Neural Networks*.

R. Schapire. 2002. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*.

S. Singh, M. Kearns, D. Litman, and M. Walker. 1999. Reinforcement learning for spoken dialogue systems. In *Proc. NIPS '99*.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. The MIT Press.

J. Tetreault and D. Litman. 2006. Comparing the utility of state features in spoken dialogue using reinforcement learning. In *NAACL*.

M. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *JAIR*, 12.

An Exploration of Eye Gaze in Spoken Language Processing for Multimodal Conversational Interfaces

Shaolin Qu **Joyce Y. Chai**
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
{qushaoli,jchai}@cse.msu.edu

Abstract

Motivated by psycholinguistic findings, we are currently investigating the role of eye gaze in spoken language understanding for multimodal conversational systems. Our assumption is that, during human machine conversation, a user's eye gaze on the graphical display indicates salient entities on which the user's attention is focused. The specific domain information about the salient entities is likely to be the content of communication and therefore can be used to constrain speech hypotheses and help language understanding. Based on this assumption, this paper describes an exploratory study that incorporates eye gaze in salience modeling for spoken language processing. Our empirical results show that eye gaze has a potential in improving automated language processing. Eye gaze is subconscious and involuntary during human machine conversation. Our work motivates more in-depth investigation on eye gaze in attention prediction and its implication in automated language processing.

1 Introduction

Psycholinguistic experiments have shown that eye gaze is tightly linked to human language processing. Eye gaze is one of the reliable indicators of what a person is "thinking about" (Henderson and Ferreira, 2004). The direction of gaze carries information about the focus of the users attention (Just and Carpenter, 1976). The perceived visual context influences spoken word recognition and mediates syntactic processing (Tanenhaus et al., 1995; Roy and Mukherjee, 2005). In addition, directly before speaking a word, the eyes move to the mentioned object (Griffin and Bock, 2000).

Motivated by these psycholinguistic findings, we are currently investigating the role of eye gaze in spoken language understanding during human machine conversation. Through multimodal interfaces, a user can look at a graphic display and converse with the system at the same time. Our assumption is that, during human machine conversation, a user's eye gaze on the graphical display can indicate salient entities on which the user's attention is focused. The specific domain information about the salient entities is likely linked to the content of communication and therefore can be used to constrain speech hypotheses and influence language understanding.

Based on this assumption, we carried out an exploration study where eye gaze information is incorporated in a salience model to tailor a language model for spoken language processing. Our preliminary results show that eye gaze can be useful in improving spoken language processing and the effect of eye gaze varies among different users. Because eye gaze is subconscious and involuntary in human machine conversation, our work also motivates systematic investigations on how eye gaze contributes to attention prediction and its implications in automated language processing.

2 Related Work

Eye gaze has been mainly used in human machine interaction as a pointing mechanism in direct manipulation interfaces (Jacob, 1990; Jacob, 1995; Zhai et al., 1999), as a facilitator in computer supported human human communication (Velichkovsky, 1995; Vertegaal, 1999); or as an additional modality during speech or multimodal communication (Starker and Bolt, 1990; Campana et al., 2001; Kaur et al., 2003; Qvarfordt and Zhai, 2005). This last area of investigation is more related to our work.

In the context of speech and multimodal communication, studies have shown that speech and eye gaze integration patterns can be modeled reliably for users. For example, by studying patterns of eye gaze and speech in the phrase “move it there”, researchers found that the gaze fixation closest to the intended object begins, with high probability, before the beginning of the word “move” (Kaur et al., 2003). Recent work has also shown that eye gaze has a potential to improve reference resolution in a spoken dialog system (Campana et al., 2001). Furthermore, eye gaze also plays an important role in managing dialog in conversational systems (Qvarfordt and Zhai, 2005).

Salience modeling has been used in both natural language and multimodal language processing. Linguistic salience describes entities with their accessibility in a hearer’s memory and their implications in language production and interpretation. Linguistic salience modeling has been used for language interpretations such as reference resolution (Huls et al., 1995; Eisenstein and Christoudias, 2004). Visual salience measures how much attention an entity attracts from a user based on its visual properties. Visual salience can tailor users’ referring expressions and thus can be used for multimodal reference resolution (Kehler, 2000). Our recent work has also investigated salience modeling based on deictic gestures to improve spoken language understanding (Chai and Qu, 2005; Qu and Chai, 2006).

3 Data Collection

We conducted user studies to collect speech and eye gaze data. In the experiments, a static 3D bedroom scene was shown to the user. The system verbally asked a user a list of questions one at a time about the bedroom and the user answered the questions by speaking to the system. Fig.1 shows the 14 questions in the experiments. The user’s speech was recorded through an open microphone and the user’s eye gaze was captured by an Eye Link II eye tracker. From 7 users’ experiments, we collected 554 utterances with a vocabulary of 489 words. Each utterance was transcribed and annotated with entities that were being talked about in the utterance.

1 Describe this room.
2 What do you like/dislike about the arrangement?
3 Describe anything in the room that seems strange to you.
4 Is there a bed in this room?
5 How big is the bed?
6 Describe the area around the bed.
7 Would you make any changes to the area around the bed?
8 Describe the left wall.
9 How many paintings are there in this room?
10 Which is your favorite painting?
11 Which is your least favorite painting?
12 What is your favorite piece of furniture in the room?
13 What is your least favorite piece of furniture in the room?
14 How would you change this piece of furniture to make it better?

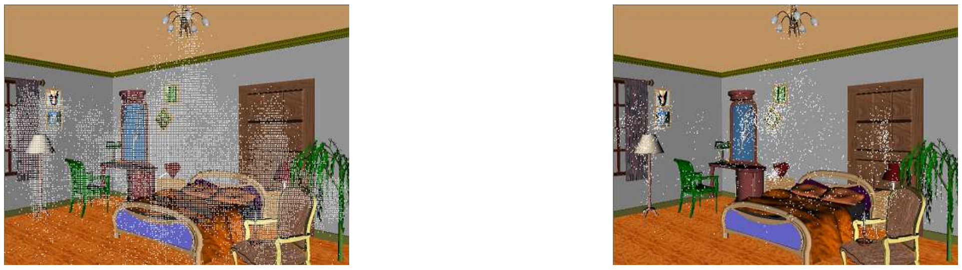
Figure 1: Questions for users in experiments

The collected raw gaze data consists of the screen coordinates of each gaze point sampled at 4 ms. As shown in Fig.2a, this raw data is not very useful for identifying fixated entities. The raw gaze data are processed to eliminate invalid and saccadic gaze points, leaving only pertinent eye fixations. Invalid gaze points occur when users look off the screen. Saccadic gaze points occur during ballistic eye movements between fixations. Vision studies have shown that no visual processing occurs during saccades (i.e., saccadic suppression). It is well known that eyes do not stay still, but rather make small, frequent jerky movements. In order to best determine fixation locations, nearby gaze points are averaged together to identify fixations. The processed eye gaze fixations can be seen in Fig.2b.

Fig.3 shows an excerpt of the collected speech and gaze fixation with fixated entities. In the speech stream, each word starts at a particular timestamp. In the gaze stream, each gaze fixation f has a starting timestamp t_f and a duration T_f. Gaze fixations can have different durations. An entity e on the graphical display is fixated by gaze fixation f if the area of e contains the fixation point of f. One gaze fixation can fall on multiple entities or no entity.

4 Salience Driven Language Modeling

Our goal is to use the domain specific information about the salient entities on a graphical display, as indicated by the user’s eye gaze, to help recognition of the user’s utterances. In particular, we incorporate this salient domain information in speech recognition via salience driven language modeling.



(a) Raw gaze points (b) Processed gaze fixations

Figure 2: Gaze fixations on a scene

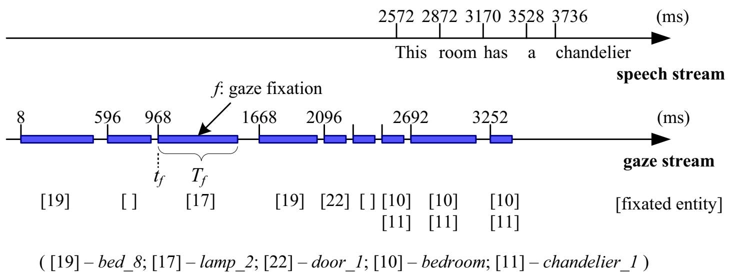


Figure 3: An excerpt of speech and gaze stream data

We first briefly introduce speech recognition. The task of speech recognition is to, given an observed spoken utterance O, find the word sequence W^* such that $W^* = \arg\max_W p(O|W)p(W)$, where $p(O|W)$ is the acoustic model and $p(W)$ is the language model. The acoustic model provides the probability of observing the acoustic features given hypothesized word sequences while the language model provides the probability of a word sequence. The language model is represented as:

$$p(W) = p(w_1^n) = \prod_{k=1}^{n} p(w_k|w_1^{k-1}) \tag{1}$$

Using first-order Markov assumption, the above language model can be approximated by a bigram model:

$$p(w_1^n) = \prod_{k=1}^{n} p(w_k|w_{k-1}) \tag{2}$$

In the following sections, we first introduce the salience modeling based on eye gaze, then present how the gaze-based salience models can be used to tailor language models.

4.1 Gaze-based Salience Modeling

We first define a gaze fixation set $F_{t_0}^{t_0+T}(e)$, which contains all gaze fixations that fall on entity e within a time window $t_0 \sim (t_0+T)$:

$$F_{t_0}^{t_0+T}(e) = \{f | f \text{ falls on } e \text{ within } t_0 \sim (t_0+T)\}$$

We model gaze-based salience in two ways.

4.1.1 Gaze Salience Model 1

Salience model 1 is based on the assumption that when an entity has more gaze fixations on it than other entities, this entity is more likely attended by the user and thus has higher salience:

$$p_{t_0,T}(e) = \frac{\#\text{elements in } F_{t_0}^{t_0+T}(e)}{\sum_e(\#\text{elements in } F_{t_0}^{t_0+T}(e))} \tag{3}$$

Here, $p_{t_0,T}(e)$ tells how likely it is that the user is focusing on entity e within time period $t_0 \sim (t_0+T)$ based on how many gaze fixations are on e among all gaze fixations that fall on entities within $t_0 \sim (t_0+T)$.

4.1.2 Gaze Salience Model 2

Salience model 2 is based on the assumption that when an entity has longer gaze fixations on it than other entities, this entity is more likely attended by the user and thus has higher salience:

$$p_{t_0,T}(e) = \frac{D_{t_0}^{t_0+T}(e)}{\sum_e D_{t_0}^{t_0+T}(e)} \tag{4}$$

where

$$D_{t_0}^{t_0+T}(e) = \sum_{f \in F_{t_0}^{t_0+T}(e)} T_f \tag{5}$$

Here, $p_{t_0,T}(e)$ tells how likely it is that the user is focusing on entity e within time period $t_0 \sim (t_0+t)$

based on how long e has been fixated by gaze fixations among the overall time length of all gaze fixations that fall on entities within $t_0 \sim (t_0 + T)$.

4.2 Salience Driven N-gram Model

Salience models can be incorporated in different language models, such as bigram models, class-based bigram models, and probabilistic context free grammar. Among these language models, the salience driven bigram model based on deictic gesture has been shown to achieve best performance on speech recognition (Qu and Chai, 2006). In our initial investigation of gaze-based salience, we incorporate the gaze-based salience in a bigram model.

The salience driven bigram probability is given by:

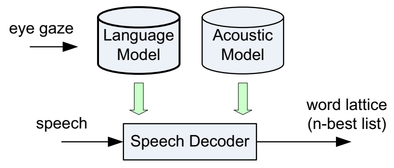
$$p_s(w_i|w_{i-1}) = (1-\lambda)p(w_i|w_{i-1}) + \lambda \sum_e p(w_i|w_{i-1}, e)p_{t_0,T}(e) \quad (6)$$

where $p_{t_0,T}(e)$ is the salience distribution as modeled in equations (3) and (4). In applying the salience driven bigram model for speech recognition, we set t_0 as the starting timestamp of the utterance and T as the duration of the utterance. The priming weight λ decides how much the original bigram probability will be tailored by the salient entities indicated by eye gaze. Currently, we set $\lambda = 0.67$ empirically. We also tried learning the priming weight with an EM algorithm. However, we found out that the learned priming weight performed worse than the empirical one in our experiments. This is probably due to insufficient development data. Bigram probabilities $p(w_i|w_{i-1})$ were estimated by the maximum likelihood estimation using Katz's backoff method (Katz, 1987) with a frequency cutoff of 1. The same method was used to estimate $p(w_i|w_{i-1}, e)$ from the users' utterance transcripts with entity annotation of e.

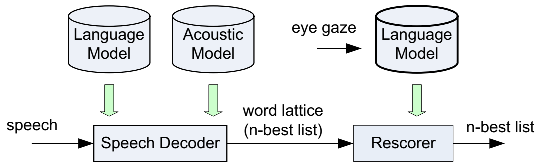
5 Application of Salience Driven LMs

The salience driven language models can be integrated into speech processing in two stages: an **early stage** before a word lattice (n-best list) is generated (Fig.4a), or in a **late stage** where the word lattice (n-best list) is post-processed (Fig.4b).

For the early stage integration, the gaze-based salience driven language model is used together with



(a) Early stage integration



(b) Late stage integration

Figure 4: Integration of gaze-based salience driven language model in speech processing

the acoustic model to generate the word lattice, typically by Viterbi search.

For the late stage integration, the gaze-based salience driven language model is used to rescore the word lattice generated by a speech recognizer with a basic language model not involving salience modeling. A* search can be applied to find the n-best paths in the word lattice.

6 Evaluation

The evaluations were conducted on data collected from user studies (Sec. 3). We evaluated the gaze-based salience driven bigram models when applied for speech recognition at early and late stages.

6.1 Evaluation Results

Users' speech was first segmented, then recognized by the CMU Sphinx-4 speech recognizer using different language models. Evaluation was done by a 14-fold cross validation. We compare the performances of the early and late applications of two gaze-based salience driven language models:

- S-Bigram1 – salience driven language model based on salience modeling 1 (Sec. 4.1.1)
- S-Bigram2 – salience driven language model based on salience modeling 2 (Sec. 4.1.2)

Table 1 and Table 2 show the results of early and late application of the salience driven language models based on eye gaze. We can see that all word error rates (WERs) are high. In the experiments, users were instructed to only answer systems questions one by one. There was no flow of a real conversation. In this setting, users were more free to express themselves than in the situation where users believed they were conversing with a machine. Thus, we observe much longer sentences that often contain disfluencies. Here is one example:

> System: "*How big is the bed?*"
> User: "*I would to have to offer a guess that the bed, if I look the chair that's beside it [pause] in a relative angle to the bed, it's probably six feet long, possibly, or shorter, slightly shorter.*"

The high WER was mainly caused by the complexity and disfluencies of users' speech. Poor speech recording quality is another reason for the bad recognition performance. It was found that the trigram model performed worse than the bigram model in the experiment. This is probably due to the sparseness of trigrams in the corpus. The amount of data available is too small considering the vocabulary size.

Language Model	Lattice-WER	WER
Bigram	0.613	0.707
Trigram	0.643	0.719
S-Bigram 1	0.605	0.690
S-Bigram 2	0.604	0.689

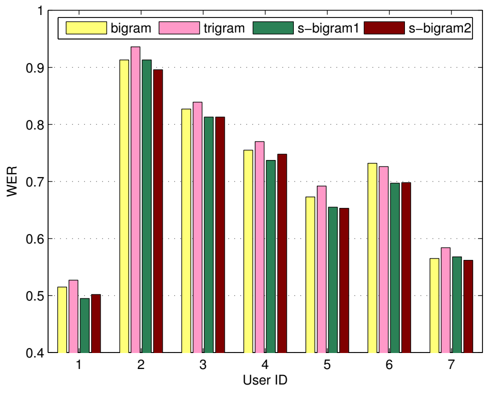
Table 1: WER of early application of LMs

Language Model	Lattice-WER	WER
S-Bigram 1	0.643	0.709
S-Bigram 2	0.643	0.710

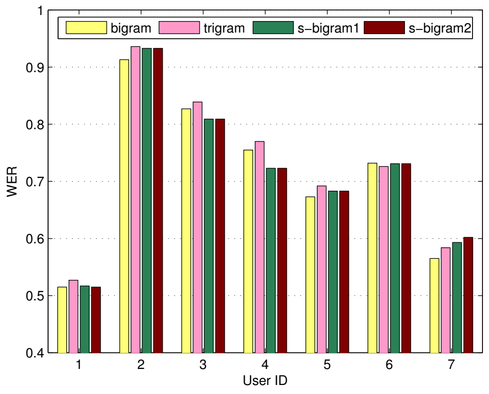
Table 2: WER of late application of LMs

The S-Bigram1 and S-Bigram2 achieved similar results in both early application (Table 1) and late application (Table 2). In early application, the S-Bigram1 model performed better than the trigram model ($t = 5.24$, $p < 0.001$, one-tailed) and the bigram model ($t = 3.31$, $p < 0.001$, one-tailed). The S-Bigram2 model also performed better than the trigram model ($t = 5.15$, $p < 0.001$, one-tailed) and the bigram model ($t = 3.33$, $p < 0.001$, one-tailed) in early application. In late application, the S-Bigram1 model performed better than the trigram model ($t = 2.11$, $p < 0.02$, one-tailed), so did the S-Bigram2 model ($t = 1.99$, $p < 0.025$, one-tailed). However, compared to the bigram model, the S-Bigram1 model did not change the recognition performance significantly ($t = 0.38$, N.S., two-tailed) in late application, neither did the S-Bigram2 model ($t = 0.50$, N.S., two-tailed).

We also compare performances of the salience driven language models for individual users. In early application (Fig.5a), both the S-Bigram1 and the S-Bigram2 model performed better than the baselines of the bigram and trigram models for all users except user 2 and user 7. T-tests have shown that these are significant improvements. For user 2, the S-Bigram1 model achieved the same WER as the bigram model. For user 7, neither of the salience driven language models improved recognition compared to the bigram model. In late application (Fig.5b), only for user 3 and user 4, both salience driven language models performed better than the baselines of the bigram and trigram models. These improvements have also been confirmed by t-tests as significant.



(a) WER of early application



(b) WER of Late application

Figure 5: WERs of LMs for individual users

Comparing early and late application of the salience driven language models, it is observed that early application performed better than late application for all users except user 3 and user 4. T-tests have confirmed that these differences are significant.

It is interesting to see that the effect of gaze-based salience modeling is different among users. For two users (i.e., user 3 and user 4), the gaze-based salience driven language models consistently outperformed the bigram and trigram models in both early application and late application. However, for some other users (e.g., user 7), this is not the case. In fact, the gaze-based salience driven language models performed worse than the bigram model. This observation indicates that during language production, a user's eye gaze is voluntary and unconscious. This is different from deictic gesture, which is more intentionally delivered by a user. Therefore, incorporating this "unconscious" mode of modality in salience modeling requires more in-depth research on the role of eye gaze in attention prediction during multimodal human computer interaction.

6.2 Discussion

Gaze-based salience driven language models are built on the assumption that when a user is fixating on an entity, the user is saying something related to the entity. With this assumption, gaze-based salience driven language models have the potential to improve speech recognition by biasing the speech decoder to favor the words that are consistent with the entity indicated by the user's eye gaze, especially when the user's utterance contains words describing unique characteristics of the entity. These particular characteristics could be the entity's name or physical properties (e.g., color, material, size).

Utterance: "a tree growing from the floor"

Gaze salience:
p(bedroom) = 0.2414 p(plant_willow) = 0.2414
p(chair_soft) = 0.2414 p(door_1) = 0.1378
p(bed_8) = 0.1378

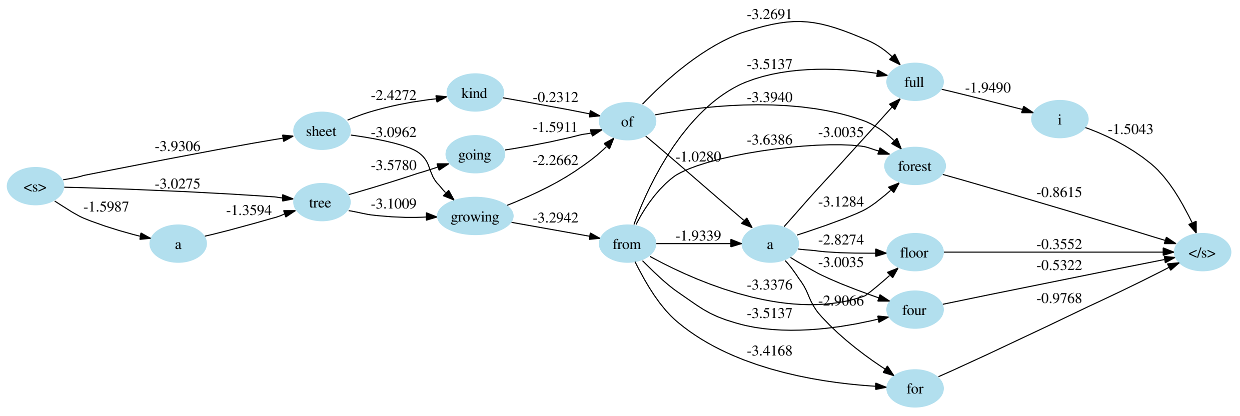
Bigram n-best list:
sheet growing from a four
sheet growing from a for
sheet growing from a floor
. . .

S-Bigram2 n-best list:
a tree growing from the floor
a tree growing from the for
a tree growing from the floor a
. . .

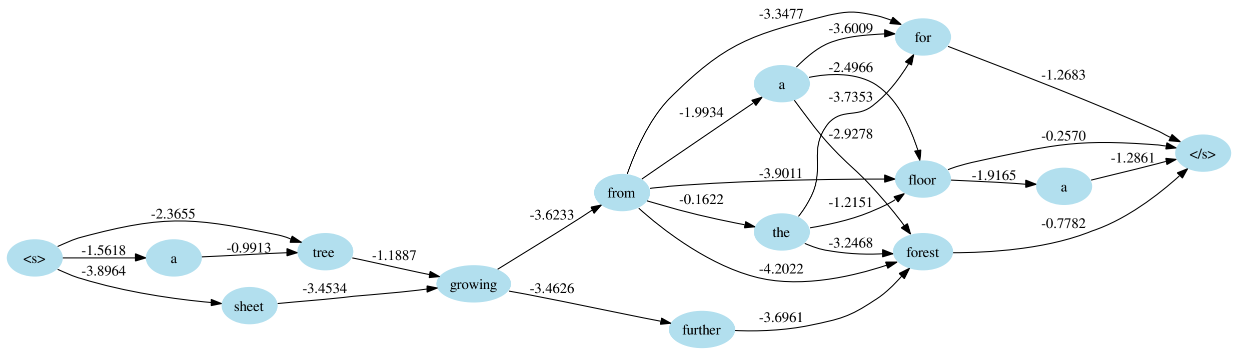
Figure 6: N-best lists of utterance "*a tree growing from the floor*"

Fig.6 shows an example where the S-Bigram2 model in early application improved recognition of the utterance "*a tree growing from the floor*". In this example, the user's gaze fixations accompanying the utterance resulted in a list of candidate entities with fixating probabilities (cf. Eqn. (4)), among which entities *bedroom* and *plant_willow* were assigned higher probabilities. Two n-best lists, the Bigram n-best list and the S-Bigram2 n-best list, were generated by the speech recognizer when the bigram model and the S-Bigram2 model were applied separately. The speech recognizer did not get the correct recognition when the bigram model was used, but got the correct result when the S-Bigram2 model was used.

Fig.7a and 7b show the word lattices of the utterance generated by the speech recognizer using the bigram model and the S-Bigram2 model respectively. The n-best lists in Fig.6 were generated from those word lattices. In the word lattices, each path going from the start node <s> to the end node </s> forms a recognition hypothesis. The bigram probabilities along the edges are in the logarithm of base 10. In the bigram case, the path "<s> a tree" has a higher language score (summation of bigram probabilities along the path) than "<s> sheet", and "a floor" has a higher language score than "a full". However, these correct paths "<s> a tree" and "a floor" (not exactly correct, but better than "a full") do not appear in the best hypothesis in the resulting n-best list. This is because the system tries to find an overall best hypothesis by considering both language and acoustic score. Because of the noisy speech, the incorrect hypotheses may happen to have higher acoustic confidence than the correct ones. After tailoring the bigram model with gaze salience, the salient entity *plant_willow* significantly increases the probability of "a tree" (from -1.3594 to -0.9913) and "tree growing" (from -3.1009 to -1.1887), while it decreases the probability of "sheet growing" (from -3.0962 to -3.4534). This probability change is made by the entity conditional probability $p(w_i|w_{i-1}, e)$ in tailoring of bigram by salience (cf. Eqn. (6)). Probability $p(w_i|w_{i-1}, e)$, trained from the annotated utterances, reflects what words are more likely to be spoken by a user while talking about an entity e. The increased probabilities of "a tree" and "tree growing" show that word "tree" appears more likely than "sheet" when the user is talking about entity



(a) Word lattice with bigram model



(b) Word lattice with S-Bigram 2

Figure 7: Word lattices of utterance "*a tree growing from the floor*"

"*plant_willow*. This is in accordance with our common sense. Likewise, the salient entity *bedroom*, of which *floor* is a component, makes the probability of the correct hypothesis "the floor" much higher than other hypotheses ("the for" and "the forest"). These enlarged language score differences make the correct hypotheses "a tree" and "the floor" win out in the searching procedure despite the noisy speech.

Utterance: "I like the picture with like a forest in it"

Gaze salience:
p(bedroom) = 0.5960 p(chandelier_1) = 0.4040

Bigram n-best list:
and i eight that picture rid like got five
and i eight that picture rid identifiable
and i eight that picture rid like got forest
. . .

S-Bigram2 n-best list:
and i that bedroom it like upside
and i that bedroom it like a five
and i that bedroom it like a forest
. . .

Figure 8: N-best lists of utterance "*I like the picture with like a forest in it*"

Unlike the active input mode of deictic gesture, eye gaze is a passive input mode. The salience information indicated by eye gaze is not as reliable as the one indicated by deictic gesture. When the salient entities indicated by eye gaze are not the true entities the user is referring to, the salience driven language model can worsen speech recognition. Fig.8 shows an example where the S-Bigram2 model in early application worsened the recognition of a user's utterance "*I like the picture with like a forest in it*" because of wrong salience information. In this example, the user was talking about a picture entity *picture_bamboo*. However, this entity was not salient, only entities *bedroom* and *chandelier_1* were salient. As a result, the recognition with the S-Bigram2 model becomes worse than the baseline. The correct word "picture" is missing and the wrong word "bedroom" appears in the result.

The failure to identify the actual referred entity *picture_bamboo* as salient in the above example can also be caused by the visual properties of entities. Smaller entities on the screen are harder to be fixated by eye gaze than larger entities. To address this issue, more reliable salience modeling that takes into account the visual features is needed.

7 Conclusion

This paper presents an empirical exploration of incorporating eye gaze in spoken language processing via salience driven language modeling. Our preliminary results have shown the potential of eye gaze in improving spoken language processing. Nevertheless, this exploratory study is only the first step in our investigation. Many interesting research questions remain. During human machine conversation, how is eye gaze aligned with speech production? How reliable is eye gaze for attention prediction? Are there any other factors such as interface design and visual properties that will affect eye gaze behavior and therefore attention prediction? The answers to these questions will affect how eye gaze should be appropriately modeled and used for language processing.

Eye-tracking systems are no longer bulky, stationary systems that prevent natural human machine communication. Recently developed display mounted gaze-tracking systems (e.g., Tobii) are completely non-intrusive, can tolerate head motion, and provide high tracking quality. These features have been demonstrated in several successful applications (Duchowski, 2002). Integrating eye tracking with conversational interfaces is no longer beyond reach. We believe it is time to conduct systematic investigations and fully explore the additional channel provided by eye gaze in improving robustness of human machine conversation.

8 Acknowledgments

This work was supported by a Career Award IIS-0347548 and IIS-0535112 from the National Science Foundation. The authors would like to thank Zahar Prasov for his contribution on data collection and thank anonymous reviewers for their valuable comments and suggestions.

References

E. Campana, J. Baldridge, J. Dowding, B. Hockey, R. Remington, and L. Stone. 2001. Using eye movements to determine referents in a spoken dialogue system. In *Proceedings of the Workshop on Perceptive User Interface*.

J. Chai and S. Qu. 2005. A salience driven approach to robust input interpretation in multimodal conversational systems. In *Proceedings of HLT/EMNLP'05*.

A. T. Duchowski. 2002. A breath-first survey of eye tracking applications. *Behavior Research methods, Instruments, and Computers*, 33(4).

J. Eisenstein and C. M. Christoudias. 2004. A salience-based approach to gesture-speech alignment. In *Proceedings of HLT/NAACL'04*.

Z. M. Griffin and K. Bock. 2000. What the eyes say about speaking. *Psychological Science*, 11:274–279.

J. M. Henderson and F. Ferreira. 2004. *The interface of language, vision, and action: Eye movements and the visual world*. New York: Taylor & Francis.

C. Huls, E. Bos, and W. Classen. 1995. Automatic referent resolution of deictic and anaphoric expressions. *Computational Linguistics*, 21(1):59–79.

R. J. K. Jacob. 1990. What you look is what you get: Eye movement-based interaction techniques. In *Proceedings of CHI'90*.

R. J. K. Jacob. 1995. Eye tracking in advanced interface design. In W. Barfield and T. Furness, editors, *Advanced Interface Design and Virtual Environments*, pages 258–288. Oxford University Press.

M. Just and P. Carpenter. 1976. Eye fixations and cognitive processes. *Cognitive Psychology*, 8:441–480.

S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Trans. Acous., Speech and Sig. Processing*, 35(3):400–401.

M. Kaur, M. Termaine, N. Huang, J. Wilder, Z. Gacovski, F. Flippo, and C. S. Mantravadi. 2003. Where is "it"? event synchronization in gaze-speech input systems. In *Proceedings of ICMI'03*.

A. Kehler. 2000. Cognitive status and form of reference in multimodal human-computer interaction. In *Proceedings of AAAI'00*.

S. Qu and J. Chai. 2006. Salience modeling based on non-verbal modalities for spoken language understanding. In *Proceedings of ICMI'06*.

P. Qvarfordt and S. Zhai. 2005. Conversing with the user based on eye-gaze patterns. In *Proceedings of CHI'05*.

D. Roy and N. Mukherjee. 2005. Towards situated speech understanding: Visual context priming of language models. *Computer Speech and Language*, 19(2):227–248.

I. Starker and R. A. Bolt. 1990. A gaze-responsive self-disclosing display. In *Proceedings of CHI'90*.

M. K. Tanenhaus, M. J. Spivey-Knowlton, K. M. Eberhard, and J. E. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

B. M. Velichkovsky. 1995. Communicating attention-gaze position transfer in cooperative problem solving. *Pragmatics and Cognition*, 3:99–224.

R. Vertegaal. 1999. The gaze groupware system: Mediating joint attention in multiparty communication and collaboration. In *Proceedings of CHI'99*.

S. Zhai, C. Morimoto, and S. Ihde. 1999. Manual and gaze input cascaded (magic) pointing. In *Proceedings of CHI'99*.

Extracting Semantic Orientations of Phrases from Dictionary

Hiroya Takamura
Precision and Intelligence Laboratory
Tokyo Institute of Technology
takamura@pi.titech.ac.jp

Takashi Inui
Integrated Research Institute
Tokyo Institute of Technology
inui@iri.titech.ac.jp

Manabu Okumura
Precision and Intelligence Laboratory
Tokyo Institute of Technology
oku@pi.titech.ac.jp

Abstract

We propose a method for extracting semantic orientations of phrases (pairs of an adjective and a noun): positive, negative, or neutral. Given an adjective, the semantic orientation classification of phrases can be reduced to the classification of words. We construct a lexical network by connecting similar/related words. In the network, each node has one of the three orientation values and the neighboring nodes tend to have the same value. We adopt the Potts model for the probability model of the lexical network. For each adjective, we estimate the states of the nodes, which indicate the semantic orientations of the adjective-noun pairs. Unlike existing methods for phrase classification, the proposed method can classify phrases consisting of unseen words. We also propose to use unlabeled data for a seed set of probability computation. Empirical evaluation shows the effectiveness of the proposed method.

1 Introduction

Technology for affect analysis of texts has recently gained attention in both academic and industrial areas. It can be applied to, for example, a survey of new products or a questionnaire analysis. Automatic sentiment analysis enables a fast and comprehensive investigation.

The most fundamental step for sentiment analysis is to acquire the semantic orientations of words: positive or negative (desirable or undesirable). For example, the word “beautiful” is positive, while the word “dirty” is negative. Many researchers have developed several methods for this purpose and obtained good results. One of the next problems to be solved is to acquire semantic orientations of phrases, or multi-term expressions, such as “high+risk” and “light+laptop-computer”. Indeed the semantic orientations of phrases depend on context just as the semantic orientations of words do, but we would like to obtain the orientations of phrases as basic units for sentiment analysis. We believe that we can use the obtained basic orientations of phrases for affect analysis of higher linguistic units such as sentences and documents.

A computational model for the semantic orientations of phrases has been proposed by Takamura et al. (2006). However, their method cannot deal with the words that did not appear in the training data. The purpose of this paper is to propose a method for extracting semantic orientations of phrases, which is applicable also to expressions consisting of unseen words. In our method, we regard this task as the noun classification problem for each adjective; the nouns that become respectively positive (negative, or neutral) when combined with a given adjective are distinguished from the other nouns. We create a lexical network with words being nodes, by connecting two words if one of the two appears in the gloss of the other. In the network, each node has one of the three orientation values and the neighboring nodes expectedly tend to have the same value. For example, the gloss of "cost" is "a sacrifice, loss, or penalty" and these words (cost, sacrifice, loss, and penalty) have the same orientation. To capture this tendency of the network, we adopt the Potts model for the probability distribution of the lexical network. For each adjective, we estimate the states of the nodes, which indicate the semantic orientations of the adjective-noun pairs. Information from seed words is diffused to unseen nouns on the network.

We also propose a method for enlarging the seed set by using the output of an existing method for the seed words of the probability computation.

Empirical evaluation shows that our method works well both for seen and unseen nouns, and that the enlarged seed set significantly improves the classification performance of the proposed model.

2 Related Work

The semantic orientation classification of words has been pursued by several researchers. Some of them used corpora (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003), while others used dictionaries (Kobayashi et al., 2001; Kamps et al., 2004; Takamura et al., 2005; Esuli and Sebastiani, 2005).

Turney (2002) applied an internet-based technique to the semantic orientation classification of phrases, which had originally been developed for word sentiment classification. In their method, the number of hits returned by a search-engine, with a query consisting of a phrase and a seed word (e.g., "*phrase* NEAR good") is used to determine the orientation. Baron and Hirst (2004) extracted collocations with Xtract (Smadja, 1993) and classified the collocations using the orientations of the words in the neighboring sentences. Their method is similar to Turney's in the sense that cooccurrence with seed words is used. In addition to individual seed words, Kanayama and Nasukawa (2006) used more complicated syntactic patterns that were manually created. The four methods above are based on context information. In contrast, our method exploits the internal structure of the semantic orientations of phrases.

Wilson et al. (2005) worked on phrase-level semantic orientations. They introduced a polarity shifter. They manually created the list of polarity shifters. Inui (2004) also proposed a similar idea.

Takamura et al. (2006) proposed to use based on latent variable models for sentiment classification of noun-adjective pairs. Their model consists of variables respectively representing nouns, adjectives, semantic orientations, and latent clusters, as well as the edges between the nodes. The words that are similar in terms of semantic orientations, such as "risk" and "mortality" (i.e., the positive orientation emerges when they are "low"), make a cluster in their model, which can be an automated version of Inui's or Wilson et al.'s idea above. However, their method cannot do anything for the words that did not appear in the labeled training data. In this paper, we call their method the latent variable method (LVM).

3 Potts Model

If a variable can have more than two values and there is no ordering relation between the values, the network comprised of such variables is called *Potts* model (Wu, 1982). In this section, we explain the simplified mathematical model of Potts model, which is used for our task in Section 4. The Potts system has been used as a mathematical model in several applications such as image restoration (Tanaka and Morita, 1996) and rumor transmission (Liu et al., 2001).

3.1 Introduction to the Potts Model

Suppose a network consisting of nodes and weighted edges is given. States of nodes are represented by c. The weight between i and j is represented by w_{ij}.

Let $H(\mathbf{c})$ denote an energy function, which indicates a state of the whole network:

$$H(\mathbf{c}) = -\beta \sum_{ij} w_{ij}\delta(c_i, c_j) + \alpha \sum_{i \in L} -\delta(c_i, a_i), \quad (1)$$

where β is a constant called *the inverse-temperature*, L is the set of the indices for the observed variables, a_i is the state of each observed variable indexed by i, and α is a positive constant representing a weight on labeled data. Function δ returns 1 if two arguments are equal to each other, 0 otherwise. The state is penalized if c_i $(i \in L)$ is different from a_i. Using $H(\mathbf{c})$, the probability distribution of the network is represented as $P(\mathbf{c}) = \exp\{-H(\mathbf{c})\}/Z$, where Z is a normalization factor.

However, it is computationally difficult to exactly estimate the state of this network. We resort to a mean-field approximation method that is described by Nishimori (2001). In the method, $P(\mathbf{c})$ is replaced by factorized function $\rho(\mathbf{c}) = \prod_i \rho_i(c_i)$. Then we can obtain the function with the smallest value of the variational free energy:

$$\begin{aligned} F(\mathbf{c}) &= \sum_{\mathbf{c}} P(\mathbf{c})H(\mathbf{c}) - \sum_{\mathbf{c}} -P(\mathbf{c}) \log P(\mathbf{c}) \\ &= -\alpha \sum_i \sum_{c_i} \rho_i(c_i)\delta(c_i, a_i) \\ &\quad -\beta \sum_{ij} \sum_{c_i,c_j} \rho_i(c_i)\rho_j(c_j)w_{ij}\delta(c_i, c_j) \\ &\quad -\sum_i \sum_{c_i} -\rho_i(c_i) \log \rho_i(c_i). \end{aligned} \quad (2)$$

By minimizing $F(\mathbf{c})$ under the condition that $\forall i$, $\sum_{c_i} \rho_i(c_i) = 1$, we obtain the following fixed point equation for $i \in L$:

$$\rho_i(c) = \frac{\exp(\alpha\delta(c, a_i) + \beta \sum_j w_{ij}\rho_j(c))}{\sum_n \exp(\alpha\delta(n, a_i) + \beta \sum_j w_{ij}\rho_j(n))}. \quad (3)$$

The fixed point equation for $i \notin L$ can be obtained by removing $\alpha\delta(c, a_i)$ from above.

This fixed point equation is solved by an iterative computation. In the actual implementation, we represent ρ_i with a linear combination of the discrete Tchebycheff polynomials (Tanaka and Morita, 1996). Details on the Potts model and its computation can be found in the literature (Nishimori, 2001).

After the computation, we obtain the function $\prod_i \rho_i(c_i)$. When the number of classes is 2, the Potts model in this formulation is equivalent to the mean-field Ising model (Nishimori, 2001).

3.2 Relation to Other Models

This Potts model with the mean-field approximation has relation to several other models.

As is often discussed (Mackay, 2003), the minimization of the variational free energy (Equation (2)) is equivalent to the obtaining the factorized model that is most similar to the maximum likelihood model in terms of the Kullback-Leibler divergence.

The second term of Equation (2) is the entropy of the factorized function. Hence the optimization problem to be solved here is a kind of the maximum entropy model with a penalty term, which corresponds to the first term of Equation (2).

We can find a similarity also to the PageRank algorithm (Brin and Page, 1998), which has been applied also to natural language processing tasks (Mihalcea, 2004; Mihalcea, 2005). In the PageRank algorithm, the pagerank score r_i is updated as

$$r_i = (1-d) + d \sum_j w_{ij} r_j, \quad (4)$$

where d is a constant ($0 \leq d \leq 1$). This update equation consists of the first term corresponding to random jump from an arbitrary node and the second term corresponding to the random walk from the neighboring node.

Let us derive the first order Taylor expansion of Equation (3). We use the equation for $i \notin L$ and denote the denominator by Z_β, for simplicity. Since $\exp x \approx 1 + x$, we obtain

$$\begin{aligned} \rho_i(c) &= \frac{\exp(\beta \sum_j w_{ij}\rho_j(c))}{Z_\beta} \\ &\approx \frac{1 + \beta \sum_j w_{ij}\rho_j(c)}{Z_\beta} \\ &= \frac{1}{Z_\beta} + \frac{\beta}{Z_\beta} \sum_j w_{ij}\rho_j(c). \end{aligned} \quad (5)$$

Equation (5) clearly has a quite similar form as Equation (4). Thus, the PageRank algorithm can be regarded as an approximation of our model. Let us clarify the difference between the two algorithms. The PageRank is designed for two-class classification, while the Potts model can be used for an arbitrary number of classes. In this sense, the PageRank is an approximated Ising model. The PageRank is applicable to asymmetric graphs, while the theory used in this paper is based on symmetric graphs.

4 Potts Model for Phrasal Semantic Orientations

In this section, we explain our classification method, which is applicable also to the pairs consisting of an adjective and an unseen noun.

4.1 Construction of Lexical Networks

We construct a lexical network, which Takamura et al. (2005) call the gloss network, by linking two words if one word appears in the gloss of the other word. Each link belongs to one of two groups:

the same-orientation links SL and the different-orientation links DL.

If a negation word (e.g., nai, for Japanese) follows a word in the gloss of the other word, the link is a different-orientation link. Otherwise the links is a same-orientation link[1].

We next set weights $W = (w_{ij})$ to links :

$$w_{ij} = \begin{cases} \frac{1}{\sqrt{d(i)d(j)}} & (l_{ij} \in SL) \\ -\frac{1}{\sqrt{d(i)d(j)}} & (l_{ij} \in DL) \\ 0 & otherwise \end{cases}, \quad (6)$$

where l_{ij} denotes the link between word i and word j, and $d(i)$ denotes the degree of word i, which means the number of words linked with word i. Two words without connections are regarded as being connected by a link of weight 0.

4.2 Classification of Phrases

Takamura et al. (2005) used the Ising model to extract semantic orientations of words (not phrases). We extend their idea and use the Potts model to extract semantic orientations of phrasal expressions.

Given an adjective, the decision remaining to be made in classification of phrasal expressions concerns nouns. We therefore estimate the state of the nodes on the lexical network for each adjective. The nouns paring with the given adjective in the training data are regarded as seed words, which we call seen words, while the words that did not appear in the training data are referred to as unseen words.

We use the mean-field method to estimate the state of the system. If the probability $\rho_i(c)$ of a variable being positive (negative, neutral) is the highest of the three classes, then the word corresponding to the variable is classified as a positive (negative, neutral) word.

We explain the reason why we use the Potts model instead of the Ising model. While only two classes (i.e., positive and negative) can be modeled by the Ising model, three classes (i.e., positive, negative and neutral) can be modelled by the Potts model. For the semantic orientations of words, all the words are sorted in the order of the average orientation value, equivalently the probability of the word being positive. Therefore, even if the neutral class is not explicitly incorporated, we can manually determine two thresholds that define respectively the positive/neutral and negative/neutral boundaries. For the semantic orientations of phrasal expressions, however, it is impractical to manually determine the thresholds for each of the numerous adjectives. Therefore, we have to incorporate the neutral class using the Potts model.

[1]For English data, a negation should *precede* a word, in order for the corresponding link to be a different-orientation link.

For some adjectives, the semantic orientation is constant regardless of the nouns. We need not use the Potts model for those unambiguous adjectives. We thus propose the following two-step classification procedure for a given noun-adjective pair $< n, a >$.

1. if the semantic orientation of all the instances with a in L is c, then classify $< n, a >$ into c.
2. otherwise, use the Potts model.

We can also construct a probability model for each noun to deal with unseen adjectives. However, we focus on the unseen nouns in this paper, because our dataset has many more nouns than adjectives.

4.3 Hyper-parameter Prediction

The performance of the proposed method largely depends on the value of hyper-parameter β. In order to make the method more practical, we propose a criterion for determining its value.

Takamura et al. (2005) proposed two kinds of criteria. One of the two criteria is an approximated leave-one-out error rate and can be used only when a large labeled dataset is available. The other is a notion from statistical physics, that is, *magnetization*:

$$m = \sum_i \bar{x}_i / N. \quad (7)$$

At a high temperature, variables are randomly oriented (*paramagnetic phase*, $m \approx 0$). At a low temperature, most of the variables have the same direction (*ferromagnetic phase*, $m \neq 0$). It is known that at some intermediate temperature, ferromagnetic phase suddenly changes to paramagnetic phase. This phenomenon is called *phase transition*. Slightly before the phase transition, variables are locally polarized; strongly connected nodes have the same polarity, but not in a global way. Intuitively, the state of the lexical network is locally polarized.

Therefore, they calculate values of m with several different values of β and select the value just before the phase transition.

Since we cannot expect a large labeled dataset to be available for each adjective, we use not the approximated leave-one-out error rate, but the magnetization-like criterion. However, the magnetization above is defined for the Ising model. We therefore consider that the phase transition has occurred, if a certain class c begins to be favored all over the system. In practice, when the maximum of the spatial averages of the approximated probabilities $\max_c \sum_i \rho_i(c)/N$ exceeds a threshold during increasing β, we consider that the phase transition has occurred. We select the value of β slightly before the phase transition.

4.4 Enlarging Seed Word Set

We usually have only a few seed words for a given adjective. Enlarging the set of seed words will increase the classification performance. Therefore, we automatically classify unlabeled pairs by means of an existing method and use the classified instances as seeds.

As an existing classifier, we use LVM. Their model can classify instances that consist of a seen noun and a seen adjective, but are unseen as a pair. Although we could classify and use all the nouns that appeared in the training data (with an adjective which is different from the given one), we do not adopt such an alternative, because it will incorporate even non-collocating pairs such as "green+idea" into seeds, resulting in possible degradation of classification performance. Therefore, we sample unseen pairs consisting of a seen noun and a seen adjective from a corpus, classify the pairs with the latent variable model, and add them to the seed set. The enlarged seed set consists of pairs used in newspaper articles and does not include non-collocating pairs.

5 Experiments

5.1 Dataset

We extracted pairs of a noun (subject) and an adjective (predicate), from Mainichi newspaper articles (1995) written in Japanese, and annotated the pairs with semantic orientation tags : positive, neutral or negative. We thus obtained the labeled dataset consisting of 12066 pair instances (7416 different pairs). The dataset contains 4459 negative instances, 4252 neutral instances, and 3355 positive instances. The number of distinct nouns is 4770 and the number of distinct adjectives is 384. To check the inter-annotator agreement between two annotators, we calculated κ statistics, which was 0.640[2]. This value is allowable, but not quite high. However, positive-negative disagreement is observed for only 0.7% of the data. In other words, this statistics means that the task of extracting neutral examples, which has hardly been explored, is intrinsically difficult.

We should note that the judgment in annotation depends on which perspective the annotator takes; "high+salary" is positive from employee's perspective, but negative from employer's perspective. The annotators are supposed to take a perspective subjectively. Our attempt is to imitate annotator's decision. To construct a classifier that matches the decision of the average person, we also have to address how to create an average corpus. We do not pursue this issue because it is out of the scope of the paper.

As unlabeled data, we extracted approximately 65,000 pairs for each iteration of the 10-fold cross-validation, from the same news source.

The average number of seed nouns for each ambiguous adjective was respectively 104 in the labeled seed set and 264 in the labeled+unlabeled seed set. Please note that these figures are counted for only ambiguous adjectives. Usually ambiguous adjectives are more frequent than unambiguous adjectives.

5.2 Experimental Settings

We employ 10-fold cross-validation to obtain the averaged classification accuracy. We split the data such that there is no overlapping pair (i.e., any pair in the training data does not appear in the test data).

Hyperparameter α was set to 1000, which is very large since we regard the labels in the seed set is reliable. For the seed words added by the classifier, lower α can be better. Determining a good value for α is regarded as future work.

Hyperparameter β is automatically selected from

[2]Although Kanayama and Nasukawa (2006) that κ for their dataset similar to ours was 0.83, this value cannot be directly compared with our value because their dataset includes both individual words and pairs of words.

$\{0.1, 0.2, \cdots, 2.5\}$ for each adjective and each fold of the cross-validation using the prediction method described in Section 4.3.

5.3 Results

The results of the classification experiments are summarized in Table 1.

The proposed method succeeded in classifying, with approximately 65% in accuracy, those phrases consisting of an ambiguous adjective and an unseen noun, which could not be classified with existing computational models such as LVM.

Incorporation of unlabeled data improves accuracy by 15.5 points for pairs consisting of a seen noun and an ambiguous adjective, and by 3.5 points for pairs consisting of an unseen noun and an ambiguous adjective, approximately. The reason why the former obtained high increase is that pairs with an ambiguous adjective[3] are usually frequent and likely to be found in the added unlabeled dataset.

If we regard this classification task as binary classification problems where we are to classify instances into one class or not, we obtain three accuracies: 90.76% for positive, 81.75% for neutral, and 86.85% for negative. This results suggests the identification of neutral instances is relatively difficult.

Next we compare the proposed method with LVM. The latent variable method is applicable only to instance pairs consisting of an adjective and a seen noun. Therefore, we computed the accuracy for 6586 instances using the latent variable method and obtained 80.76 %. The corresponding accuracy by our method was 80.93%. This comparison shows that our method is better than or at least comparable to the latent variable method. However, we have to note that this accuracy of the proposed method was computed using the unlabeled data classified by the latent variable method.

5.4 Discussion

There are still 3320 (=12066-8746) word pairs which could not be classified, because there are no entries for those words in the dictionary. However, the main cause of this problem is word segmentation, since many compound nouns and exceedingly-subdivided morphemes are not in dictionaries. An appropriate mapping from the words found in corpus to entries of a dictionary will solve this problem. We found a number of proper nouns, many of which are not in the dictionary. By estimating a class of a proper noun and finding the words that matches the class in the dictionary, we can predict the semantic orientations of the proper noun based on the orientations of the found words.

In order to see the overall tendency of errors, we calculated the confusion matrices both for pairs of an ambiguous adjective and a seen noun, and for pairs of an ambiguous adjective and an unseen noun (Table 2). The proposed method works quite well for positive/negative classification, though it finds still some difficulty in correctly classifying neutral instances even after enhanced with the unlabeled data.

In order to qualitatively evaluate the method, we list several word pairs below. These word pairs are classified by the Potts model with the labeled+unlabeled seed set. All nouns are unseen; they did not appear in the original training dataset. Please note again that the actual data is Japanese.

positive instances

noun	adjective
cost	low
basic price	low
loss	little
intelligence	high
educational background	high
contagion	not-happening
version	new
cafe	many
salary	high
commission	low

negative instances

noun	adjective
damage	heavy
chance	little
terrorist	many
trouble	many
variation	little
capacity	small
salary	low
disaster	many
disappointment	big
knowledge	little

For example, although both "salary" and "commission" are kinds of money, our method captures

[3]Seen nouns are observed in both the training and the test datasets because they are frequent. Ambiguous adjectives are often-used adjectives such as "large", "small", "high", and "low".

Table 1: Classification accuracies (%) for various seed sets and test datasets. 'Labeled' seed set corresponds to the set of manually labeled pairs. 'Labeled+unlabeled' seed set corresponds to the union of 'labeled' seed set and the set of pairs labeled by LVM. 'Seen nouns' for test are the nouns that appeared in the training data, while 'unseen nouns' are the nouns that did not appear in the training dataset'. Please note that seen pairs are excluded from the test data. 'Unambiguous' adjectives corresponds to the pairs with an adjective which has a unique orientation in the original training dataset, while 'ambiguous' adjectives corresponds to the pairs with an adjective which has more than one orientation in the original training dataset.

seed\test	seen nouns		unseen nouns		total
labeled	68.24 (4494/6586)		73.70 (1592/2160)		69.59 (6086/8746)
	unambiguous	ambiguous	unambiguous	ambiguous	
	98.15 (1166/1188)	61.65 (3328/5398)	94.85 (736/776)	61.85 (856/1384)	
labeled+unlabeled	80.93 (5330/6586)		75.88 (1639/2160)		79.68 (6969/8746)
	unambiguous	ambiguous	unambiguous	ambiguous	
	98.15 (1166/1188)	77.14 (4164/5398)	94.85 (736/776)	65.25 (903/1384)	

Table 2: Confusion matrices of classification result with labeled+unlabeled seed set

		Potts model							
		seen nouns				unseen nouns			
		positive	neutral	negative	sum	positive	neutral	negative	sum
Gold standard	positive	964	254	60	1278	126	84	30	240
	neutral	198	1656	286	2140	60	427	104	591
	negative	39	397	1544	1980	46	157	350	553
	sum	1201	2307	1890	5398	232	668	484	1384

the difference between them; "high salary" is positive, while "low (cheap) commission" is also positive.

6 Conclusion

We proposed a method for extracting semantic orientations of phrases (pairs of an adjective and a noun). For each adjective, we constructed a Potts system, which is actually a lexical network extracted from glosses in a dictionary. We empirically showed that the proposed method works well in terms of classification accuracy.

Future work includes the following:

- We assumed that each word has a semantic orientation. However, word senses and subjectivity have strong interaction (Wiebe and Mihalcea, 2006).
- The value of α must be properly set, because lower α can be better for the seed words added by the classifier,
- To address word-segmentation problem discussed in Section 5.3, we can utilize the fact that the heads of compound nouns often inherit the property determining the semantic orientation when combined with an adjective.
- The semantic orientations of pairs consisting of a proper noun will be estimated from the named entity classes of the proper nouns such as person name and organization.

References

Faye Baron and Graeme Hirst. 2004. Collocations as cues to semantic orientation. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

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.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss analysis. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*, pages 617–624.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 174–181.

Takashi Inui. 2004. *Acquiring Causal Knowledge from Text Using Connective Markers*. Ph.D. thesis, Graduate School of Information Science, Nara Institute of Science and Technology.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten de Rijke. 2004. Using wordnet to measure semantic orientation of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04), volume IV*, pages 1115–1118.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'06)*, pages 355–363.

Nozomi Kobayashi, Takashi Inui, and Kentaro Inui. 2001. Dictionary-based acquisition of the lexical knowledge for p/n analysis (in Japanese). In *Proceedings of Japanese Society for Artificial Intelligence, SLUD-33*, pages 45–50.

Zhongzhu Liu, Jun Luo, and Chenggang Shao. 2001. Potts model for exaggeration of a simple rumor transmitted by recreant rumormongers. *Physical Review E*, 64:046134,1–046134,9.

David J. C. Mackay. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Mainichi. 1995. Mainichi Shimbun CD-ROM version.

Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *The Companion Volume to the Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, (ACL'04)*, pages 170–173.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the Joint Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 411–418.

Hidetoshi Nishimori. 2001. *Statistical Physics of Spin Glasses and Information Processing*. Oxford University Press.

Frank Z. Smadja. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 133–140.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2006. Latent variable models for semantic orientations of phrases. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*.

Kazuyuki Tanaka and Tohru Morita. 1996. Application of cluster variation method to image restoration problem. In *Theory and Applications of the Cluster Variation and Path Probability Methods*, pages 353–373. Plenum Press, New York.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424.

Janyce M. Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL'06)*, pages 1065–1072.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of joint conference on Human Language Technology / Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP'05)*, pages 347–354.

Fa-Yueh Wu. 1982. The potts model. *Reviews of Modern Physics*, 54(1):235–268.

Multiple Aspect Ranking using the Good Grief Algorithm

Benjamin Snyder and Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{bsnyder,regina}@csail.mit.edu

Abstract

We address the problem of analyzing multiple related opinions in a text. For instance, in a restaurant review such opinions may include food, ambience and service. We formulate this task as a multiple aspect ranking problem, where the goal is to produce a set of numerical scores, one for each aspect. We present an algorithm that jointly learns ranking models for individual aspects by modeling the dependencies between assigned ranks. This algorithm guides the prediction of individual rankers by analyzing meta-relations between opinions, such as agreement and contrast. We prove that our agreement-based joint model is more expressive than individual ranking models. Our empirical results further confirm the strength of the model: the algorithm provides significant improvement over both individual rankers and a state-of-the-art joint ranking model.

1 Introduction

Previous work on sentiment categorization makes an implicit assumption that a single score can express the polarity of an opinion text (Pang et al., 2002; Turney, 2002; Yu and Hatzivassiloglou, 2003). However, multiple opinions on related matters are often intertwined throughout a text. For example, a restaurant review may express judgment on food quality as well as the service and ambience of the restaurant. Rather than lumping these aspects into a single score, we would like to capture each aspect of the writer's opinion separately, thereby providing a more fine-grained view of opinions in the review.

To this end, we aim to predict a set of numeric ranks that reflects the user's satisfaction for each aspect. In the example above, we would assign a numeric rank from 1-5 for each of: food quality, service, and ambience.

A straightforward approach to this task would be to rank[1] the text independently for each aspect, using standard ranking techniques such as regression or classification. However, this approach fails to exploit meaningful dependencies between users' judgments across different aspects. Knowledge of these dependencies can be crucial in predicting accurate ranks, as a user's opinions on one aspect can influence his or her opinions on others.

The algorithm presented in this paper models the dependencies between different labels via *the agreement relation*. The agreement relation captures whether the user equally likes all aspects of the item or whether he or she expresses different degrees of satisfaction. Since this relation can often be determined automatically for a given text (Marcu and Echihabi, 2002), we can readily use it to improve rank prediction.

The Good Grief model consists of a ranking model for each aspect as well as an agreement model which predicts whether or not all rank aspects are

[1]In this paper, *ranking* refers to the task of assigning an integer from 1 to k to each instance. This task is sometimes referred to as "ordinal regression" (Crammer and Singer, 2001) and "rating prediction" (Pang and Lee, 2005).

Proceedings of NAACL HLT 2007, pages 300–307,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

equal. The Good Grief decoding algorithm predicts a set of ranks – one for each aspect – which maximally satisfy the preferences of the individual rankers and the agreement model. For example, if the agreement model predicts consensus but the individual rankers select ranks $\langle 5, 5, 4 \rangle$, then the decoder decides whether to trust the the third ranker, or alter its prediction and output $\langle 5, 5, 5 \rangle$ to be consistent with the agreement prediction. To obtain a model well-suited for this decoding, we also develop a joint training method that conjoins the training of multiple aspect models.

We demonstrate that the agreement-based joint model is more expressive than individual ranking models. That is, every training set that can be perfectly ranked by individual ranking models for each aspect can also be perfectly ranked with our joint model. In addition, we give a simple example of a training set which cannot be perfectly ranked without agreement-based joint inference. Our experimental results further confirm the strength of the Good Grief model. Our model significantly outperforms individual ranking models as well as a state-of-the-art joint ranking model.

2 Related Work

Sentiment Classification Traditionally, categorization of opinion texts has been cast as a binary classification task (Pang et al., 2002; Turney, 2002; Yu and Hatzivassiloglou, 2003; Dave et al., 2003). More recent work (Pang and Lee, 2005; Goldberg and Zhu, 2006) has expanded this analysis to the ranking framework where the goal is to assess review polarity on a multi-point scale. While this approach provides a richer representation of a single opinion, it still operates on the assumption of one opinion per text. Our work generalizes this setting to the problem of analyzing multiple opinions – or multiple aspects of an opinion. Since multiple opinions in a single text are related, it is insufficient to treat them as separate single-aspect ranking tasks. This motivates our exploration of a new method for joint multiple aspect ranking.

Ranking The ranking, or ordinal regression, problem has been extensivly studied in the Machine Learning and Information Retrieval communities. In this section we focus on two online ranking methods which form the basis of our approach. The first is a model proposed by Crammer and Singer (2001). The task is to predict a rank $y \in \{1, ..., k\}$ for every input $\mathbf{x} \in \mathbb{R}^n$. Their model stores a weight vector $\mathbf{w} \in \mathbb{R}^n$ and a vector of increasing boundaries $b_0 = -\infty \leq b_1 \leq ... \leq b_{k-1} \leq b_k = \infty$ which divide the real line into k segments, one for each possible rank. The model first scores each input with the weight vector: $score(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. Finally, the model locates $score(\mathbf{x})$ on the real line and returns the appropriate rank as indicated by the boundaries. Formally, the model returns the rank r such that $b_{r-1} \leq score(\mathbf{x}) < b_r$. The model is trained with the Perceptron Ranking algorithm (or "PRank algorithm"), which reacts to incorrect predictions on the training set by updating the weight and boundary vectors. The PRanking model and algorithm were tested on the `EachMovie` dataset with a separate ranking model learned for each user in the database.

An extension of this model is provided by Basilico and Hofmann (2004) in the context of collaborative filtering. Instead of training a separate model for each user, Basilico and Hofmann train a joint ranking model which shares a set of boundaries across all users. In addition to these shared boundaries, user-specific weight vectors are stored. To compute the score for input $\mathbf{x}$ and user i, the weight vectors for *all* users are employed:

$$score_i(\mathbf{x}) = \mathbf{w}[i] \cdot \mathbf{x} + \sum_j sim(i, j)(\mathbf{w}[j] \cdot \mathbf{x}) \quad (1)$$

where $0 \leq sim(i, j) \leq 1$ is the cosine similarity between users i and j, computed on the entire training set. Once the score has been computed, the prediction rule follows that of the PRanking model. The model is trained using the PRank algorithm, with the exception of the new definition for the scoring function.[2] While this model shares information between the different ranking problems, it fails to explicitly model relations between the rank predictions. In contrast, our algorithm uses an agreement model to learn such relations and inform joint predictions.

[2]In the notation of Basilico and Hofmann (2004), this definition of $score_i(\mathbf{x})$ corresponds to the kernel $K = (K_U^{id} + K_U^{co}) \oplus K_X^{at}$.

3 The Algorithm

The goal of our algorithm is to find a rank assignment that is consistent with predictions of individual rankers and the agreement model. To this end, we develop the Good Grief decoding procedure that minimizes the dissatisfaction (*grief*) of individual components with a joint prediction. In this section, we formally define the grief of each component, and a mechanism for its minimization. We then describe our method for joint training of individual rankers that takes into account the Good Grief decoding procedure.

3.1 Problem Formulation

In an *m-aspect ranking problem*, we are given a training sequence of instance-label pairs $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^t, \mathbf{y}^t), ...$ Each instance $\mathbf{x}^t$ is a feature vector in $\mathbb{R}^n$ and the label $\mathbf{y}^t$ is a vector of m ranks in $\mathcal{Y}^m$, where $\mathcal{Y} = \{1, .., k\}$ is the set of possible ranks. The i^{th} component of $\mathbf{y}^t$ is the rank for the i^{th} aspect, and will be denoted by $y[i]^t$. The goal is to learn a mapping from instances to rank sets, $H : \mathcal{X} \rightarrow \mathcal{Y}^m$, which minimizes the distance between predicted ranks and true ranks.

3.2 The Model

Our *m-aspect ranking model* contains $m+1$ components: $(\langle \mathbf{w}[1], \mathbf{b}[1]\rangle, ..., \langle \mathbf{w}[m], \mathbf{b}[m]\rangle, \mathbf{a})$. The first m components are individual ranking models, one for each aspect, and the final component is the agreement model. For each aspect $i \in 1...m$, $\mathbf{w}[i] \in \mathbb{R}^n$ is a vector of weights on the input features, and $\mathbf{b}[i] \in \mathbb{R}^{k-1}$ is a vector of boundaries which divide the real line into k intervals, corresponding to the k possible ranks. The default prediction of the aspect ranking model simply uses the ranking rule of the PRank algorithm. This rule predicts the rank r such that $b[i]_{r-1} \leq score_i(\mathbf{x}) < b[i]_r$.[3] The value $score_i(\mathbf{x})$ can be defined simply as the dot product $\mathbf{w}[i] \cdot \mathbf{x}$, or it can take into account the weight vectors for other aspects weighted by a measure of inter-aspect similarity. We adopt the definition given in equation 1, replacing the user-specific weight vectors with our aspect-specific weight vectors.

[3]More precisely (taking into account the possibility of ties): $\hat{y}[i] = \min_{r\in\{1,..,k\}}\{r : score_i(\mathbf{x}) - b[i]_r < 0\}$

The agreement model is a vector of weights $\mathbf{a} \in \mathbb{R}^n$. A value of $\mathbf{a} \cdot \mathbf{x} > 0$ predicts that the ranks of all m aspects are equal, and a value of $\mathbf{a} \cdot \mathbf{x} \leq 0$ indicates disagreement. The absolute value $|\mathbf{a} \cdot \mathbf{x}|$ indicates the confidence in the agreement prediction.

The goal of the decoding procedure is to predict a joint rank for the m aspects which satisfies the individual ranking models as well as the agreement model. For a given input $\mathbf{x}$, the individual model for aspect i predicts a default rank $\hat{y}[i]$ based on its feature weight and boundary vectors $\langle \mathbf{w}[i], \mathbf{b}[i]\rangle$. In addition, the agreement model makes a prediction regarding rank consensus based on $\mathbf{a} \cdot \mathbf{x}$. However, the default aspect predictions $\hat{y}[1] \ldots \hat{y}[m]$ may not accord with the agreement model. For example, if $\mathbf{a} \cdot \mathbf{x} > 0$, but $\hat{y}[i] \neq \hat{y}[j]$ for some $i, j \in 1...m$, then the agreement model predicts complete consensus, whereas the individual aspect models do not.

We therefore adopt a joint prediction criterion which simultaneously takes into account *all* model components – individual aspect models as well as the agreement model. For each possible prediction $\mathbf{r} = (r[1], ..., r[m])$ this criterion assesses the level of *grief* associated with the i^{th}-aspect ranking model, $g_i(\mathbf{x}, r[i])$. Similarly, we compute the grief of the agreement model with the joint prediction, $g_a(\mathbf{x}, \mathbf{r})$ (both g_i and g_a are defined formally below). The decoder then predicts the m ranks which minimize the overall grief:

$$H(\mathbf{x}) = \arg\min_{\mathbf{r}\in\mathcal{Y}^m} \left[g_a(\mathbf{x}, \mathbf{r}) + \sum_{i=1}^{m} g_i(\mathbf{x}, r[i]) \right] \quad (2)$$

If the default rank predictions for the aspect models, $\hat{\mathbf{y}} = (\hat{y}[1], ..., \hat{y}[m])$, are in accord with the agreement model (both indicating consensus or both indicating contrast), then the grief of all model components will be zero, and we simply output $\hat{\mathbf{y}}$. On the other hand, if $\hat{\mathbf{y}}$ indicates disagreement but the agreement model predicts consensus, then we have the option of predicting $\hat{\mathbf{y}}$ and bearing the grief of the agreement model. Alternatively, we can predict some consensus $\mathbf{y}'$ (i.e. with $y'[i] = y'[j], \forall i, j$) and bear the grief of the component ranking models. The decoder H chooses the option with lowest overall grief.[4]

[4]This decoding criterion assumes that the griefs of the com-

Now we formally define the measures of *grief* used in this criterion.

Aspect Model Grief We define the grief of the i^{th}-aspect ranking model with respect to a rank r to be the smallest magnitude correction term which places the input's score into the r^{th} segment of the real line:

$$\begin{gathered} g_i(\mathbf{x}, r) = \min |c| \\ \text{s.t.} \\ b[i]_{r-1} \leq score_i(\mathbf{x}) + c < b[i]_r \end{gathered}$$

Agreement Model Grief Similarly, we define the grief of the agreement model with respect to a joint rank $\mathbf{r} = (r[1], \ldots, r[m])$ as the smallest correction needed to bring the agreement score into accord with the agreement relation between the individual ranks $r[1], \ldots, r[m]$:

$$\begin{gathered} g_a(\mathbf{x}, \mathbf{r}) = \min |c| \\ \text{s.t.} \\ \mathbf{a} \cdot \mathbf{x} + c > 0 \land \forall i, j \in 1...m : r[i] = r[j] \\ \lor \\ \mathbf{a} \cdot \mathbf{x} + c \leq 0 \land \exists i, j \in 1...m : r[i] \neq r[j] \end{gathered}$$

3.3 Training

Ranking models Pseudo-code for Good Grief training is shown in Figure 1. This training algorithm is based on PRanking (Crammer and Singer, 2001), an online perceptron algorithm. The training is performed by iteratively ranking each training input $\mathbf{x}$ and updating the model. If the predicted rank $\hat{y}$ is equal to the true rank y, the weight and boundaries vectors remain unchanged. On the other hand, if $\hat{y} \neq y$, then the weights and boundaries are updated to improve the prediction for $\mathbf{x}$ (step 4.c in Figure 1). See (Crammer and Singer, 2001) for explanation and analysis of this update rule.

Our algorithm departs from PRanking by conjoining the updates for the m ranking models. We achieve this by using Good Grief decoding at each step throughout training. Our decoder $H(\mathbf{x})$ (from equation 2) uses *all* the aspect component models as well as the (previously trained) agreement model to determine the predicted rank for each aspect. In concrete terms, for every training instance $\mathbf{x}$, we predict the ranks of all aspects simultaneously (step 2 in Figure 1). Then, for each aspect we make a separate update based on this joint prediction (step 4 in Figure 1), instead of using the individual models' predictions.

Agreement model The agreement model $\mathbf{a}$ is assumed to have been previously trained on the same training data. An instance is labeled with a positive label if all the ranks associated with this instance are equal. The rest of the instances are labeled as negative. This model can use any standard training algorithm for binary classification such as Perceptron or SVM optimization.

3.4 Feature Representation

Ranking Models Following previous work on sentiment classification (Pang et al., 2002), we represent each review as a vector of lexical features. More specifically, we extract all unigrams and bigrams, discarding those that appear fewer than three times. This process yields about 30,000 features.

Agreement Model The agreement model also operates over lexicalized features. The effectiveness of these features for recognition of discourse relations has been previously shown by Marcu and Echihabi (2002). In addition to unigrams and bigrams, we also introduce a feature that measures the maximum contrastive distance between pairs of words in a review. For example, the presence of *"delicious"* and *"dirty"* indicate high contrast, whereas the pair *"expensive"* and *"slow"* indicate low contrast. The contrastive distance for a pair of words is computed by considering the difference in relative weight assigned to the words in individually trained PRanking models.

4 Analysis

In this section, we prove that our model is able to perfectly rank a strict superset of the training corpora perfectly rankable by m ranking models individually. We first show that if the independent ranking models can individually rank a training set perfectly, then our model can do so as well. Next, we show that our model is more expressive by providing

ponent models are comparable. In practice, we take an uncalibrated agreement model $\mathbf{a}'$ and reweight it with a tuning parameter: $\mathbf{a} = \alpha\mathbf{a}'$. The value of α is estimated using the development set. We assume that the griefs of the ranking models are comparable since they are jointly trained.

```
**Input :** $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^T, \mathbf{y}^T)$, Agreement model $\mathbf{a}$, Decoder defintion $H(\mathbf{x})$ (from equation 2).
**Initialize :** Set $\mathbf{w}[i]^1 = 0$, $b[i]^1_1, ..., b[i]^1_{k-1} = 0$, $b[i]^1_k = \infty, \forall i \in 1...m$.
**Loop :** For $t = 1, 2, ..., T$ :
  1. Get a new instance $\mathbf{x}^t \in \mathbb{R}^n$.
  2. Predict $\hat{\mathbf{y}}^t = H(\mathbf{x}; \mathbf{w}^t, \mathbf{b}^t, \mathbf{a})$ (Equation 2).
  3. Get a new label $\mathbf{y}^t$.
  4. For aspect $i = 1, ..., m$:
        If $\hat{y}[i]^t \neq y[i]^t$ update model (otherwise set $\mathbf{w}[i]^{t+1} = \mathbf{w}[i]^t$, $b[i]^{t+1}_r = b[i]^t_r$, $\forall r$):
          4.a For $r = 1, ..., k-1$ :      If $y[i]^t \leq r$ then $y[i]^t_r = -1$
                                           else $y[i]^t_r = 1$.
          4.b For $r = 1, ..., k-1$ :      If $(\hat{y}[i]^t - r)y[i]^t_r \leq 0$ then $\tau[i]^t_r = y[i]^t_r$
                                           else $\tau[i]^t_r = 0$.
          4.c Update $\mathbf{w}[i]^{t+1} \leftarrow \mathbf{w}[i]^t + (\sum_r \tau[i]^t_r)\mathbf{x}^t$.
              For $r = 1, ..., k-1$ update :     $b[i]^{t+1}_r \leftarrow b[i]^t_r - \tau[i]^t_r$.

**Output :**    $H(\mathbf{x}; \mathbf{w}^{T+1}, \mathbf{b}^{T+1}, \mathbf{a})$.
```

Figure 1: Good Grief Training. The algorithm is based on PRanking training algorithm. Our algorithm differs in the joint computation of all aspect predictions $\hat{\mathbf{y}}^t$ based on the Good Grief Criterion (step 2) and the calculation of updates for each aspect based on the joint prediction (step 4).

a simple illustrative example of a training set which can only be perfectly ranked with the inclusion of an agreement model.

First we introduce some notation. For each training instance $(\mathbf{x}^t, \mathbf{y}^t)$, each aspect $i \in 1...m$, and each rank $r \in 1...k$, define an auxiliary variable $y[i]^t_r$ with $y[i]^t_r = -1$ if $y[i]^t \leq r$ and $y[i]^t_r = 1$ if $y[i]^t > r$. In words, $y[i]^t_r$ indicates whether the *true* rank $y[i]^t$ is to the right or left of a *potential* rank r.

Now suppose that a training set $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^T, \mathbf{y}^T)$ is perfectly rankable for each aspect independently. That is, for each aspect $i \in 1...m$, there exists some ideal model $v[i]^* = (w[i]^*, b[i]^*)$ such that the signed distance from the prediction to the r^{th} boundary: $\mathbf{w}[i]^* \cdot \mathbf{x}^t - b[i]^*_r$ has the same sign as the auxiliary variable $y[i]^t_r$. In other words, the minimum margin over all training instances and ranks, $\gamma = \min_{r,t}\{(\mathbf{w}[i]^* \cdot \mathbf{x}^t - b[i]^*_r)y[i]^t_r\}$, is no less than zero.

Now for the t^{th} training instance, define an agreement auxiliary variable a^t, where $a^t = 1$ when all aspects agree in rank and $a^t = -1$ when at least two aspects disagree in rank. First consider the case where the agreement model $\mathbf{a}$ perfectly classifies all training instances: $(\mathbf{a} \cdot \mathbf{x}^t)a^t > 0, \forall t$. It is clear that Good Grief decoding with the ideal joint model $(\langle \mathbf{w}[1]^*, \mathbf{b}[1]^* \rangle, ..., \langle \mathbf{w}[m]^*, \mathbf{b}[m]^* \rangle, \mathbf{a})$ will produce the same output as the component ranking models run separately (since the grief will always be zero for the default rank predictions). Now consider the case where the training data is not linearly separable with regard to agreement classification. Define the margin of the worst case error to be $\beta = \max_t\{|(\mathbf{a} \cdot \mathbf{x}^t)| : (\mathbf{a} \cdot \mathbf{x}^t)a^t < 0\}$. If $\beta < \gamma$, then again Good Grief decoding will always produce the default results (since the grief of the agreement model will be at most β in cases of error, whereas the grief of the ranking models for any deviation from their default predictions will be at least γ). On the other hand, if $\beta \geq \gamma$, then the agreement model errors could potentially disrupt the perfect ranking. However, we need only rescale $w^* := w^*(\frac{\beta}{\gamma} + \epsilon)$ and $b^* := b^*(\frac{\beta}{\gamma} + \epsilon)$ to ensure that the grief of the ranking models will always exceed the grief of the agreement model in cases where the latter is in error. Thus whenever independent ranking models can perfectly rank a training set, a joint ranking model with Good Grief decoding can do so as well.

Now we give a simple example of a training set which can only be perfectly ranked with the addition of an agreement model. Consider a training set of four instances with two rank aspects:

$$\begin{aligned}
\langle \mathbf{x}^1, \mathbf{y}^1\rangle &= \langle (1,0,1), \quad (2,1)\rangle\\
\langle \mathbf{x}^2, \mathbf{y}^2\rangle &= \langle (1,0,0), \quad (2,2)\rangle\\
\langle \mathbf{x}^3, \mathbf{y}^3\rangle &= \langle (0,1,1), \quad (1,2)\rangle\\
\langle \mathbf{x}^4, \mathbf{y}^4\rangle &= \langle (0,1,0), \quad (1,1)\rangle
\end{aligned}$$

We can interpret these inputs as feature vectors corresponding to the presence of "good", "bad", and "but not" in the following four sentences:

The food was **good**, **but not** the ambience.
The food was **good**, and so was the ambience.
The food was **bad**, **but not** the ambience.
The food was **bad**, and so was the ambience.

We can further interpret the first rank aspect as the quality of food, and the second as the quality of the ambience, both on a scale of 1-2.

A simple ranking model which only considers the words "good" and "bad" perfectly ranks the food aspect. However, it is easy to see that no single model perfectly ranks the ambience aspect. Consider any model $\langle \mathbf{w}, \mathbf{b} = (b)\rangle$. Note that $\mathbf{w} \cdot \mathbf{x}^1 < b$ and $\mathbf{w} \cdot \mathbf{x}^2 \geq b$ together imply that $w_3 < 0$, whereas $\mathbf{w} \cdot \mathbf{x}^3 \geq b$ and $\mathbf{w} \cdot \mathbf{x}^4 < b$ together imply that $w_3 > 0$. Thus independent ranking models cannot perfectly rank this corpus.

The addition of an agreement model, however, can easily yield a perfect ranking. With $\mathbf{a} = (0, 0, -5)$ (which predicts contrast with the presence of the words "but not") and a ranking model for the ambience aspect such as $\mathbf{w} = (1, -1, 0), \mathbf{b} = (0)$, the Good Grief decoder will produce a perfect rank.

5 Experimental Set-Up

We evaluate our multi-aspect ranking algorithm on a corpus[5] of restaurant reviews available on the website `http://www.we8there.com`. Reviews from this website have been previously used in other sentiment analysis tasks (Higashinaka et al., 2006). Each review is accompanied by a set of five ranks, each on a scale of 1-5, covering food, ambience, service, value, and overall experience. These ranks are provided by consumers who wrote original reviews. Our corpus does not contain incomplete data points since all the reviews available on this website contain both a review text and the values for all the five aspects.

Training and Testing Division Our corpus contains 4,488 reviews, averaging 115 words. We randomly select 3,488 reviews for training, 500 for development and 500 for testing.

Parameter Tuning We used the development set to determine optimal numbers of training iterations for our model and for the baseline models. Also, given an initial uncalibrated agreement model $\mathbf{a}'$, we define our agreement model to be $\mathbf{a} = \alpha\mathbf{a}'$ for an appropriate scaling factor α. We tune the value of α on the development set.

Corpus Statistics Our training corpus contains 528 among $5^5 = 3025$ possible rank sets. The most frequent rank set $\langle 5, 5, 5, 5, 5\rangle$ accounts for 30.5% of the training set. However, no other rank set comprises more than 5% of the data. To cover 90% of occurrences in the training set, 227 rank sets are required. Therefore, treating a rank tuple as a single label is not a viable option for this task. We also find that reviews with full agreement across rank aspects are quite common in our corpus, accounting for 38% of the training data. Thus an agreement-based approach is natural and relevant.

A rank of 5 is the most common rank for all aspects and thus a prediction of all 5's gives a MAJORITY baseline and a natural indication of task difficulty.

Evaluation Measures We evaluate our algorithm and the baseline using *ranking loss* (Crammer and Singer, 2001; Basilico and Hofmann, 2004). Ranking loss measures the average distance between the true rank and the predicted rank. Formally, given N test instances $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^N, \mathbf{y}^N)$ of an m-aspect ranking problem and the corresponding predictions $\hat{\mathbf{y}}^1, ..., \hat{\mathbf{y}}^N$, ranking loss is defined as $\sum_{t,i} \frac{|y[i]^t - \hat{y}[i]^t|}{mN}$. Lower values of this measure correspond to a better performance of the algorithm.

6 Results

Comparison with Baselines Table 1 shows the performance of the Good Grief training algorithm GG TRAIN+DECODE along with various baselines, including the simple MAJORITY baseline mentioned in section 5. The first competitive baseline, PRANK, learns a separate ranker for each aspect using the PRank algorithm. The second competitive baseline, SIM, shares the weight vectors across aspects using a similarity measure (Basilico and Hofmann, 2004).

[5]Data and code used in this paper are available at `http://people.csail.mit.edu/bsnyder/naacl07`

	Food	Service	Value	Atmosphere	Experience	**Total**
MAJORITY	0.848	1.056	1.030	1.044	1.028	1.001
PRANK	0.606	0.676	0.700	0.776	0.618	0.675
SIM	0.562	0.648	0.706	0.798	0.600	0.663
GG DECODE	0.544	0.648	0.704	0.798	0.584	0.656
GG TRAIN+DECODE	**0.534**	**0.622**	**0.644**	**0.774**	**0.584**	**0.632**
GG ORACLE	0.510	0.578	0.674	0.694	0.518	0.595

Table 1: Ranking loss on the test set for variants of Good Grief and various baselines.

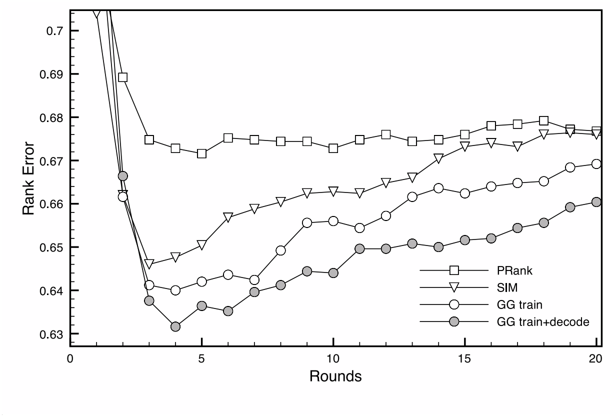


Figure 2: Rank loss for our algorithm and baselines as a function of training round.

Both of these methods are described in detail in Section 2. In addition, we consider two variants of our algorithm: GG DECODE employs the PRank training algorithm to independently train all component ranking models and only applies Good Grief decoding at test time. GG ORACLE uses Good Grief training and decoding but in both cases is given perfect knowledge of whether or not the true ranks all agree (instead of using the trained agreement model).

Our model achieves a rank error of 0.632, compared to 0.675 for PRANK and 0.663 for SIM. Both of these differences are statistically significant at $p < 0.002$ by a Fisher Sign Test. The gain in performance is observed across all five aspects. Our model also yields significant improvement ($p < 0.05$) over the decoding-only variant GG DECODE, confirming the importance of joint training. As shown in Figure 2, our model demonstrates consistent improvement over the baselines across all the training rounds.

Model Analysis We separately analyze our performance on the 210 test instances where all the target ranks agree and the remaining 290 instances where there is some contrast. As Table 2 shows, we outperform the PRANK baseline in both cases. However on the consensus instances we achieve a relative reduction in error of 21.8% compared to only a 1.1% reduction for the other set. In cases of consensus, the agreement model can guide the ranking models by reducing the decision space to five rank sets. In cases of disagreement, however, our model does not provide sufficient constraints as the vast majority of ranking sets remain viable. This explains the performance of GG ORACLE, the variant of our algorithm with perfect knowledge of agreement/disagreement facts. As shown in Table 1, GG ORACLE yields substantial improvement over our algorithm, but most of this gain comes from consensus instances (see Table 2).

	Consensus	Non-consensus
PRANK	0.414	0.864
GG TRAIN+DECODE	0.324	0.854
GG ORACLE	0.281	0.830

Table 2: Ranking loss for our model and PRANK computed separately on cases of actual consensus and actual disagreement.

We also examine the impact of the agreement model accuracy on our algorithm. The agreement model, when considered on its own, achieves classification accuracy of 67% on the test set, compared to a majority baseline of 58%. However, those instances with high confidence $|\mathbf{a} \cdot \mathbf{x}|$ exhibit substantially higher classification accuracy. Figure 3 shows the performance of the agreement model as a function of the confidence value. The 10% of the data with highest confidence values can be classified by

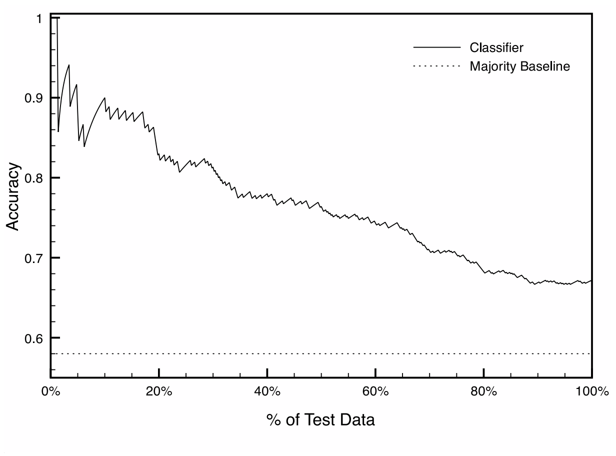


Figure 3: Accuracy of the agreement model on subsets of test instances with highest confidence $|\mathbf{a} \cdot \mathbf{x}|$.

the agreement model with 90% accuracy, and the third of the data with highest confidence can be classified at 80% accuracy.

This property explains why the agreement model helps in joint ranking even though its overall accuracy may seem low. Under the Good Grief criterion, the agreement model's prediction will only be enforced when its grief outweighs that of the ranking models. Thus in cases where the prediction confidence ($|\mathbf{a}\cdot\mathbf{x}|$) is relatively low,[6] the agreement model will essentially be ignored.

7 Conclusion and Future Work

We considered the problem of analyzing multiple related aspects of user reviews. The algorithm presented jointly learns ranking models for individual aspects by modeling the dependencies between assigned ranks. The strength of our algorithm lies in its ability to guide the prediction of individual rankers using rhetorical relations between aspects such as agreement and contrast. Our method yields significant empirical improvements over individual rankers as well as a state-of-the-art joint ranking model.

Our current model employs a single rhetorical relation – agreement vs. contrast – to model dependencies between different opinions. As our analysis shows, this relation does not provide sufficient constraints for non-consensus instances. An avenue for future research is to consider the impact of additional rhetorical relations between aspects. We also plan to theoretically analyze the convergence properties of this and other joint perceptron algorithms.

[6]What counts as "relatively low" will depend on both the value of the tuning parameter α and the confidence of the component ranking models for a particular input $\mathbf{x}$.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865) and the Microsoft Research Faculty Fellowship. Thanks to Michael Collins, Pawan Deshpande, Jacob Eisenstein, Igor Malioutov, Luke Zettlemoyer, and the anonymous reviewers for helpful comments and suggestions. Thanks also to Vasumathi Raman for programming assistance. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF.

References

J. Basilico, T. Hofmann. 2004. Unifying collaborative and content-based filtering. In *Proceedings of the ICML*, 65–72.

K. Crammer, Y. Singer. 2001. Pranking with ranking. In *NIPS*, 641–647.

K. Dave, S. Lawrence, D. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW*, 519–528.

A. B. Goldberg, X. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of HLT/NAACL workshop on TextGraphs*, 45–52.

R. Higashinaka, R. Prasad, M. Walker. 2006. Learning to generate naturalistic utterances using reviews in spoken dialogue systems. In *Proceedings of COLING/ACL*, 265–272.

D. Marcu, A. Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of ACL*, 368–375.

B. Pang, L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 115–124.

B. Pang, L. Lee, S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, 79–86.

P. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classsification of reviews. In *Proceedings of the ACL*, 417–424.

H. Yu, V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*, 129–136.

Extracting Appraisal Expressions

Kenneth Bloom and **Navendu Garg** and **Shlomo Argamon**
Computer Science Department
Illinois Institute of Technology
10 W. 31st St.
Chicago, IL 60616
{kbloom1,gargnav,argamon}@iit.edu

Abstract

Sentiment analysis seeks to characterize opinionated or evaluative aspects of natural language text. We suggest here that *appraisal expression extraction* should be viewed as a fundamental task in sentiment analysis. An *appraisal expression* is a textual unit expressing an evaluative stance towards some target. The task is to find and characterize the evaluative attributes of such elements. This paper describes a system for effectively extracting and disambiguating adjectival appraisal expressions in English outputting a generic representation in terms of their evaluative function in the text. Data mining on appraisal expressions gives meaningful and non-obvious insights.

1 Introduction

Sentiment analysis, which seeks to analyze opinion in natural language text, has grown in interest in recent years. Sentiment analysis includes a variety of different problems, including: sentiment classification techniques to classify reviews as positive or negative, based on bag of words (Pang et al., 2002) or positive and negative words (Turney, 2002; Mullen and Collier, 2004); classifying sentences in a document as either subjective or objective (Riloff and Wiebe, 2003; Pang and Lee, 2004); identifying or classifying appraisal targets (Nigam and Hurst, 2004); identifying the source of an opinion in a text (Choi et al., 2005), whether the author is expressing the opinion, or whether he is attributing the opinion to someone else; and developing interactive and visual opinion mining methods (Gamon et al., 2005; Popescu and Etzioni, 2005). Much of this work has utilized the fundamental concept of 'semantic orientation', (Turney, 2002); however, sentiment analysis still lacks a 'unified field theory'.

We propose in this paper that a fundamental task underlying many of these formulations is the extraction and analysis of *appraisal expressions*, defined as those structured textual units which express an evaluation of some object. An appraisal expression has three main components: an *attitude* (which takes an evaluative stance about an object), a *target* (the object of the stance), and a *source* (the person taking the stance) which may be implied.

The idea of appraisal extraction is a generalization of problem formulations developed in earlier works. Mullen and Collier's (2004) notion of classifying appraisal terms using a multidimensional set of attributes is closely tied to the definition of an appraisal expression, which is classified along several dimensions. In previous work (Whitelaw et al., 2005), we presented a related technique of finding opinion phrases, using a multidimensional set of attributes and modeling the semantics of modifiers in these phrases. The use of multiple text classifiers by Wiebe and colleagues (Wilson et al., 2005; Wiebe et al., 2004) for various kinds of sentiment classification can also be viewed as a sentence-level technique for analyzing appraisal expressions. Nigam and Hurst's (2004) work on detecting opinions about a certain topic presages our notion of connecting attitudes to targets, while Popescu and Etzioni's (2005) opinion mining technique also fits well into our framework.

In this paper we describe a system for extracting adjectival appraisal expressions, based on a hand-built lexicon, a combination of heuristic shallow parsing and dependency parsing, and expectation-maximization word sense disambiguation. Each extracted appraisal expression is represented as a set of feature values in terms of its evaluative function in the text. We have applied this system to two domains of texts: product reviews, and movie reviews. Manual evaluation of the extraction shows our system to work well, as well as giving some directions for improvement. We also show how straightforward data mining can give users very useful information about public opinion.

2 Appraisal Expressions

We define an *appraisal expression* to be an elementary linguistic unit that conveys an attitude of some kind towards some target. An *appraisal expression* is defined to comprise a *source*, an *attitude*, and a *target*, each represented by various attributes. For example, in 'I found the movie quite monotonous', the speaker (the *Source*) expresses a negative *Attitude* ('quite monotonous') towards 'the movie' (the *Target*). Note that attitudes come in different types; for example, 'monotonous' describes an inherent quality of the Target, while 'loathed' would describe the emotional reaction of the Source.

Attitude may be expressed through nouns, verbs, adjectives and metaphors. Extracting all of this information accurately for all of these types of appraisal expressions is a very difficult problem. We therefore restrict ourselves for now to *adjectival appraisal expressions* that are each contained in a single sentence. Additionally, we focus here only on extracting and analyzing the attitude and the target, but not the source. Even with these restrictions, we obtain interesting results (Sec. 7).

2.1 Appraisal attributes

Our method is grounded in Appraisal Theory, developed by Martin and White (2005), which analyzes the way opinion is expressed. Following Martin and White, we define:

Attitude type is type of appraisal being expressed—one of *affect*, *appreciation*, or *judgment* (Figure 1). *Affect* refers to an emotional state (e.g., 'happy', 'angry'), and is the most explicitly subjective type of appraisal. The other two types express evaluation of external entities, differentiating between intrinsic *appreciation* of object properties (e.g., 'slender', 'ugly') and social *judgment* (e.g., 'heroic', 'idiotic').

Orientation is whether the attitude is *positive* ('good') or *negative* ('bad').

Force describes the intensity of the appraisal. Force is largely expressed via modifiers such as 'very' (increased force), or 'slightly' (decreased force), but may also be expressed lexically, for example 'greatest' vs. 'great' vs. 'good'.

Polarity of an appraisal is *marked* if it is scoped in a polarity marker (such as 'not'), or *unmarked* otherwise. Other attributes of appraisal are affected by negation; e.g., 'not good' also has the opposite orientation from 'good'.

Target type is a domain-dependent semantic type for the target. This attribute takes on values from a domain-dependent taxonomy, representing important (and easily extractable) distinctions between targets in the domain.

- Attitude Type
  - Appreciation
    - Composition
      - Balance: *consistent, discordant, ...*
      - Complexity: *elaborate, convoluted, ...*
    - Reaction
      - Impact: *amazing, compelling, dull, ...*
      - Quality: *beautiful, elegant, hideous, ...*
    - Valuation: *innovative, profound, inferior, ...*
  - Affect: *happy, joyful, furious, ...*
  - Judgment
    - Social Esteem
      - Capacity: *clever, competent, immature, ...*
      - Tenacity: *brave, hard-working, foolhardy, ...*
      - Normality: *famous, lucky, obscure, ...*
    - Social Sanction
      - Propriety: *generous, virtuous, corrupt, ...*
      - Veracity: *honest, sincere, sneaky, ...*

Figure 1: The Attitude Type taxonomy, with examples of adjectives from the lexicon.

2.2 Target taxonomies

Two domain-dependent target type taxonomies are shown in Figure 2. In both, the primary distinction is between a direct naming of a kind of "Thing" or a deictic/pronominal reference (e.g., "those" or "it"), since the system does not currently rely on coreference resolution. References are further divided into references to the writer/reader ('interactants') and to other people or objects.

The Thing subtrees for the two domains differ somewhat. In the movie domain, Things such as 'this movie', 'Nicholas Cage', or 'cinematography', are classified into six main categories: movies (the one being reviewed, or another one), people

- Movie Target Type
  - Movie Thing
    - Any Movie
      - This Movie
      - Other Movie
    - Movie Person
      - Real Person. . .
      - Character
    - Movie Aspect. . .
    - Company
    - Marketing
  - Reference
    - Interactant
      - First Person
      - Second Person
    - Other
      - Third Person
      - Deictic

- Product Target Type
  - Product Thing
    - Any Product
      - This Product
      - Other Product
    - Product Part
      - Integral
      - Replaceable
    - Experience
    - Company
    - Marketing
    - Support
  - Reference
    - Interactant
      - First Person
      - Second Person
    - Other
      - Third Person
      - Deictic

Figure 2: Target taxonomies for movie and product reviews.

(whether characters, or real people involved in making the film), aspects of the movie itself (its plot, special effects, etc.), the companies involved in making it, or aspects of marketing the movie (such as trailers). For target Things in product reviews, we replace 'Movie Person' and 'Movie Aspect' by 'Product Part' with two subcategories: 'Integral', for parts of the product itself (e.g., wheels or lenses), and 'Replaceable', for parts or supplies meant to be periodically replaced (e.g., batteries or ink cartridges). The categories of 'Support', for references to aspects of customer support, and 'Experience' for things associated with the experience of using the product (such as 'pictures' or 'resolution', were also added.

3 Appraisal Extraction

In our system, appraisal extraction runs in several independent stages. First, the appraisal extractor finds appraisal expressions by finding the chunks of text that express attitudes and targets. Then, it links each attitude group found to a target in the text. Finally, it uses a probabilistic model to determine which attitude type should be assigned when attitude chunks were ambiguous.

3.1 Chunking

The chunker is based on our earlier work (Whitelaw et al., 2005), which finds attitude groups and targets using a hand-built lexicon (Sec. 4). This lexicon contains head adjectives (which specify values for the attributes attitude type, force, polarity, and orientation), and appraisal modifiers (which specify transformations to the four attributes). Some head adjectives are ambiguous, having multiple entries in the lexicon with different attribute values. In all cases, different entries for a given word have different attitude types. If the head adjective is ambiguous, multiple groups are created, to be disambiguated later. See our previous work (Whitelaw et al., 2005) for a discussion of the technique.

Target groups are found by matching phrases in the lexicon with corresponding phrases in the text and assigning the target type listed in the lexicon.

3.2 Linking

After finding attitude groups and candidate targets, the system links each attitude to a target. Each sentence is parsed to a dependency representation, and a ranked list of linkage specifications is used to look for paths in the dependency tree connecting some word in the source to some word in the target. Such linkage specifications are hand-constructed, and manually assigned priorities, so that when two linkage specifications match, only the highest priority specification is used. For example, the two highest priority linkage specifications are:

1. target $\xrightarrow{nsubj} x \xleftarrow{dobj} y \xleftarrow{amod}$ attitude

2. attitude $\xrightarrow{amod}$ target

The first specification selects the subject of a sentence where the appraisal modifies a noun in the predicate, for example 'The Matrix' in 'The Matrix is a good movie'. The second selects the noun modified by an adjective group, for example 'movie' in 'The Matrix is a good movie'.

If no linkage is found connecting an attitude to a candidate target, the system goes through the linkage specifications again, trying to find any word in the sentence connected to the appraisal group by a known linkage. The selected word is assigned the generic category of *movie thing* or *product thing* (depending on the domain of the text). If no linkage is found at all, the system assigns the default category *movie thing* or *product thing*, assuming that there is an appraised thing that couldn't be found using the given linkage specifications.

3.3 Disambiguation

After linkages are made, this information is used to disambiguate multiple senses that may be present in a given appraisal expression. Most cases are unambiguous, but in some cases two, or occasionally even three, senses are possible. We bootstrap from the unambiguous cases, using a probabilistic model, to resolve the ambiguities. The attitude type places some grammatical/semantic constraints on the clause. Two key constraints are the syntactic relation with the target (which can differentiate *affect* from the other types of appraisal), and whether the target type has consciousness (which helps differentiate *judgment* and *affect* from *appreciation*). To capture these constraints, we model the probability of a given attitude type being correct, given the target type and the linkage specification used to connect the attitude to the target, as follows.

The correct attitude type of an appraisal expression is modeled by a random variable A, the set of all attitude types in the system is denoted by $\mathcal{A}$, and a specific attitude type is denoted by a. As described above, other attributes besides attitude type may also vary between word senses, but attitude type always changes between word senses, so when the system assigns a probability to an attitude type, it is assigning that probability to the whole word sense.

We denote the linkage type used in a given appraisal expression by L, the set of all possible linkages as $\mathcal{L}$, and a specific linkage type by l. Note that the first attempt with a linkage specification (to find a chunked target) is considered to be different from the second attempt with the same linkage specification (which attempts to find any word). Failure to find an applicable linkage rule is considered as yet another 'linkage' for the probability model. Since our system uses 29 different linkage specifications, there are a total of 59 different possible linkages types.

The target type of a given appraisal expression is denoted by T, the set of all target types by $\mathcal{T}$, and a specific target type by t. We consider an expression to have a given target type $T = t$ only if that is its specific target type; if its target type is a descendant of t, then its target type is not t in the model. $\mathcal{E}$ denotes the set of all extracted appraisal expressions. The term exp denotes a specific expression.

Our goal is to estimate, for each appraisal expression exp in the corpus, the probability of its attitude type being a, given the expression's target type t and linkage type l

$$P(A = a|exp) = P(A = a|T = t, L = l)$$

To do this, we define a model M of this probability, and then estimate the maximum likelihood model using Expectation-Maximization.

We model $P_M(A = a|T = t, L = l)$ by first applying Bayes' theorem:

$$P_M(A = a|T = t, L = l) =$$

$$\frac{P_M(T = t, L = l|A = a)P_M(A = a)}{P_M(T = t, L = l)}$$

Assuming conditional independence of target type and linkage, this becomes:

$$\frac{P_M(T = t|A = a)P_M(L = l|A = a)P_M(A = a)}{P_M(T = t)P_M(L = l)}$$

M's parameters thus represent the conditional and marginal probabilities on this right-hand-side.

Given a set of (possibly ambiguous) appraisal expressions $\mathcal{E}$ identified by chunking and linkage detection, we seek the maximum likelihood model

$$M^* = \arg\max_M \prod_{exp \in \mathcal{E}} \prod_{a \in \mathcal{A}} M(A = a|exp)$$

M^* will be our best estimate of P, given the processed data in a given corpus. The system estimates M^* using an implementation of Expectation-Maximization over the entire corpus. The highest-probability attitude type (hence sense) according to M is then chosen for each appraisal expression.

4 The Lexicon

As noted above, attitude groups were identified via a domain-independent lexicon of appraisal adjectives, adverbs, and adverb modifiers. [1] For the movie domain, appraised things were identified based on a manually constructed lexicon containing generic movie words, as well as automatically constructed lexicons of proper names specific to each movie being reviewed. For each product type considered, we manually constructed a lexicon containing generic product words; we did not find it necessary to construct product-specific lexicons.

[1]All of the lexicons used in the paper can be found at `http://lingcog.iit.edu/arc/appraisal_lexicon_2007a.tar.gz`

For adjectival attitudes, we used the lexicon developed we developed in our previous work (Whitelaw et al., 2005) on appraisal. We reviewed the entire lexicon to determine its accuracy and made numerous improvements.

Generic target lexicons were constructed by starting with a small sample of the kind of reviews that the lexicon would apply to. We examined these manually to find generic words referring to appraised things to serve as seed terms for the lexicon and used WordNet (Miller, 1995) to suggest additional terms to add to the lexicon.

Since movie reviews often refer to the specific contents of the movie under review by proper names (of actors, the director, etc.), we also automatically constructed a *specific target lexicon* for each movie in the corpus, based on lists of actors, characters, writers, directors, and companies listed for the film at `imdb.com`. Each such specific lexicon was only used for processing reviews of the movie it was generated for, so the system had no specific knowledge of terms related to other movies during processing.

5 Corpora

We evaluated our appraisal extraction system on two corpora. The first is the standard publicly available collection of movie reviews constructed by Pang and Lee (2004). This standard testbed consists of 1000 positive and 1000 negative reviews, taken from the IMDb movie review archives[2]. Reviews with 'neutral' scores (such as three stars out of five) were removed by Pang and Lee, giving a data set with only clearly positive and negative reviews. The average document length in this corpus is 764 words, and 1107 different movies are reviewed.

The second corpus is a collection of user product reviews taken from `epinions.com` supplied in 2004 for research purposes by Amir Ashkenazi of Shopping.Com. The base collection contains reviews for three types of products: baby strollers, digital cameras, and printers. Each review has a numerical rating (1–5); based on this, we labeled positive and negative reviews in the same way as Pang and Lee did for the movie reviews corpus. The products corpus has 15162 documents, averaging 442 words long. This comprises 11769 positive documents, 1420 neutral documents, and 1973 negative documents. There are 905 reviews of strollers, 5778 reviews of ink-jet printers and 8479 reviews of digital cameras, covering 516 individual products.

Each document in each corpus was preprocessed into individual sentences, lower-cased, and tokenized. We used an implementation of Brill's (1992) part-of-speech tagger to find adjectives and modifiers; for parsing, we used the Stanford dependency parser (Klein and Manning, 2003).

[2]See `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

6 Evaluating Extraction

We performed two manual evaluations on the system. The first was to evaluate the overall accuracy of the entire system. The second was to specifically evaluate the accuracy of the probabilistic disambiguator.

6.1 Evaluating Accuracy

We evaluated randomly selected appraisal expressions for extraction accuracy on a number of binary measures. This manual evaluation was performed by the first author.We evaluated interrater reliability between this rater and another author on 200 randomly selected appraisal expressions (100 on each corpus). The first rater rated an additional 120 expressions (60 for each corpus), and combined these with his ratings for interrater reliability to compute system accuracy, for a total of 320 expressions (160 for each corpus). The (binary) rating criteria were as follows. Relating to the appraisal group:

APP Does the expression express appraisal at all?

ARM If so, does the appraisal group have all relevant modifiers?

HEM Does the appraisal group include extra modifiers? (Results are shown negated, so that higher numbers are better.)

Relating to the target:

HT If there is appraisal, is there an identifiable target (even if the system missed it)?

FT If there is appraisal, did the system identify some target? (Determined automatically.)

RT If so, is it the correct one?

Relating to the expression's attribute values (if it expresses appraisal):

Att Is the attitude type assigned correct?

Ori Is the orientation assigned correct?

Pol Is the polarity assigned correct?

Tar Is the target type assigned correct?

Pre Is the target type the most precise value in the taxonomy for this target?

Table 1: System accuracy at evaluated tasks. 95% confidence one-proportion z-intervals are reported.

Measure	Movies	Products	Combined
APP	86% ± 3%	81% ± 3%	83% ± 2%
ARM	94% ± 2%	95% ± 2%	95% ± 1%
¬ HEM	99% ± 1%	100%	99.6% ± 0.4%
HT	91% ± 2%	97% ± 2%	94% ± 1%
FT	96% ± 2%	94% ± 2%	95% ± 1%
RT	77% ± 4%	73% ± 4%	75% ± 3%
Att	78% ± 4%	80% ± 4%	79% ± 2%
Ori	95% ± 2%	95% ± 2%	94% ± 1%
Pol	97% ± 1%	96% ± 2%	97% ± 1%
Tar	84% ± 3%	86% ± 3%	85% ± 2%
Pre	70% ± 4%	77% ± 4%	73% ± 3%

Table 2: Interrater reliability of manual evaluation. 95% confidence intervals are reported.

Measure	Movies	Products	Combined
APP	71% ± 9%	87% ± 7%	79% ± 6%
ARM	95% ± 5%	91% ± 6%	93% ± 4%
¬ HEM	98% ± 3%	100%	99% ± 1%
HT	97% ± 4%	99% ± 3%	98% ± 3%
FT	N/A	N/A	N/A
RT	94% ± 6%	97% ± 4%	96% ± 4%
Att	79% ± 10%	86% ± 8%	83% ± 6%
Ori	93% ± 6%	94% ± 5%	93% ± 4%
Pol	96% ± 4%	94% ± 5%	95% ± 4%
Tar	94% ± 6%	90% ± 7%	91% ± 5%
Pre	86% ± 10%	90% ± 8%	88% ± 6%

Results are given in Table 1, and interrater reliability is given in Table 2. In nearly all cases agreement percentages are above 80%, indicating good inter-rater consensus. Regarding precision, we note that most aspects of extraction seem to work quite well. The area of most concern in the system is precision of target classification. This may be improved with further development of the target lexicons to classify more terms to specific leaves in the target type hierarchy. The other area of concern is the **APP** test, which encountered difficulties when a word could be used as appraisal in some contexts, but not in others, particularly when an appraisal word appeared as a nominal classifier.

6.2 Evaluating Disambiguation

The second experiment evaluated the accuracy of EM in disambiguating the attitude type of appraisal expressions. We evaluated the same number of expressions as used for the overall accuracy experiment (100 used for interrater reliability and accuracy, plus 60 used only for accuracy on each corpus), each having two or more word senses, presenting all of the attitude types possible for each appraisal expression, as well as a 'none of the above' and a 'not appraisal' option, asking the rater to select which one applied to the selected expression in context.

Baseline disambiguator accuracy, if the computer were to simply pick randomly from the choices specified in the lexicon is 48% for both corpora. Interrater agreement was 80% for movies and 73% for products (taken over 100 expressions from each corpus.)

Considering just those appraisal expressions which the raters decided were appraisal, the disambiguator achieved 58% accuracy on appraisal expressions from the movies corpus and 56% accuracy on the products corpus. Further analysis of the results of the disambiguator shows that most of the errors occur when the target type is the generic category *thing* which occurs when the target is not in the target lexicon. Performance on words recognized as having more specific target types is better: 68% for movies, and 59% for products. This indicates that specific target type is an important indicator of attitude type.

7 Opinion Mining

We (briefly) demonstrate the usefulness of appraisal expression extraction by using it for opinion mining. In opinion mining, we find large numbers of reviews and perform data mining to determine which aspects of a product people like or dislike, and in which ways. To do this, we search for association rules describing the appraisal features that can be found in a single appraisal expression. We generally look for rules that contain attitude type, orientation, thing type, and a product name, when these rules occur more frequently than expected.

The idea is similar to Agrawal and Srikant's (1995) notion of generalized association rules. We treat each appraisal expression as a transaction, with the attributes of attitude type, orientation, polarity, force, and thing type, as well as the document attributes product name, product type, and document classification (based on the number of stars the reviewer gave the product). We use CLOSET+ (Wang et al., 2003) to find all of the frequent closed itemsets in the data, with a support greater than or equal to 20 occurrences. Let $\langle b, a_1, a_2, \ldots a_n \rangle$ or $\langle b, A \rangle$ denote the contents of an itemset, and $c(\langle b, A \rangle)$ denote the support for this itemset. For a given item b, $\pi(b)$ denotes its immediate parent its value taxonomy, or 'root' for flat sets.

Table 3: The most interesting specific rules for products.

Int.	b Product Name		A Attitude	Target Type	Orientation	Polarity	Doc. class
45.7	**Peg Perego Pliko Matic (1)**	$\Leftarrow$	quality	this-product	positive	unmarked	
42.8	Lexmark Color JetPrinter 1100	$\Leftarrow$	reaction	this-product	negative	unmarked	neg
41.9	Peg Perego Milano XL	$\Leftarrow$	reaction	this-product	positive	unmarked	pos
41.1	Peg Perego Pliko Matic	$\Leftarrow$	reaction	this-product	positive	unmarked	
40.8	Peg Perego Milano XL	$\Leftarrow$	quality	this-product	positive	unmarked	pos
37.5	Peg Perego Milano XL	$\Leftarrow$	reaction	this-product	positive	unmarked	
37.1	Peg Perego Milano XL	$\Leftarrow$	quality	this-product	positive	unmarked	
36.3	**Agfa ePhoto Smile (2)**	$\Leftarrow$	reaction	experience	negative	unmarked	neg
36.0	**Agfa ePhoto Smile (2)**	$\Leftarrow$	reaction	experience	negative		neg
33.9	KB Gear KG-JC3S Jamcam	$\Leftarrow$	quality	experience	negative		neg

Table 4: The most interesting oppositional rules for products.

Int.	b Product Name		A Attitude	Target	Orient.	Polarity	Doc. class
31.6	**Lexmark Color JetPrinter 1100 (3)**	$\Leftarrow$	reaction	this-product	positive		neg
31.5	Lexmark Color JetPrinter 1100	$\Leftarrow$	quality	this-product	positive		neg
29.5	Lexmark Color JetPrinter 1100	$\Leftarrow$	reaction	this-product	positive	unmarked	neg
29.2	Lexmark Color JetPrinter 1100	$\Leftarrow$	quality	this-product	positive	unmarked	neg
28.9	Lexmark Color JetPrinter 1100	$\Leftarrow$	appreciation	this-product	positive		neg

For each item set, we collect rules $\langle b, A\rangle$ and compute their interestingness relative to the itemset $\langle \pi(b), A\rangle$. Interestingness is defined as follows:

$$Int = \frac{P(A|b)}{P(A|\pi(b))} = \frac{c(\langle b, A\rangle) \times c(\langle \pi(b)\rangle)}{c(\langle \pi(b), A\rangle) \times c(\langle b\rangle)}$$

Int is the relative probability of finding the child itemset in an appraisal expression, compared to finding it in a parent itemset. Values greater than 1 indicate that the child itemset appears more frequently than we would expect.

We applied two simple filters to the output, to help find more meaningful results. **Specificity** requires that b be a product name, and that attitude type and thing type be sufficiently deep nodes in the hierarchy to describe something specific. (For example, 'product thing' gives no real information about what part of the product is being appraised.) **Opposition** chooses rules with a different rating than the review as a whole, that is, document classification is the opposite of appraisal orientation. The filter also ensures that thing type is sufficiently specific, as with specificity, and requires that b be a product name.

We present the ten most 'interesting' rules from each filter, for the products corpus. Rules from the specificity filter are shown in Table 3 and rules from the opposition filter are shown in Table 4. We consider the meaning of some of these rules.

The first specificity rule **(1)** describes a typical example of users who like the product very well overall. An example sentence that created this rule says 'Not only is it an *excellent stroller*, because of it's [sic] size it even doubled for us as a portable crib.'

The specificity rules for the Agfa ePhoto Smile Digital Camera **(2)** are an example of the kind of rule we expect to see when bad user experience contributes to bad reviews. The text of the reviews that gave these rules quite clearly convey that users were not happy specifically with the photo quality.

In the oppositional rules for the Lexmark Color JetPrinter 1100 **(3)**, we see that users made positive comments about the product overall, while nevertheless giving the product a negative review. Drilling down into the text, we can see some examples of reviews like 'On the surface it looks like a *good printer* but it has many flaws that cause it to be frustrating.'

8 Conclusions

We have presented a new task, *appraisal expression extraction*, which, we suggest, is a fundamental tasks for sentiment analysis. Shallow parsing based on a set of appraisal lexicons, together with sparse use of syntactic dependencies, can be used to effectively address the subtask of extracting adjectival appraisal expressions. Indeed, straightforward data mining applied to appraisal expressions can yield insights into public opinion as expressed in patterns of evaluative language in a corpus of product reviews.

Immediate future work includes extending the approach to include other types of appraisal expressions, such as where an attitude is expressed via a noun or a verb. In this regard, we will be examining extension of existing methods for automatically building lexicons of positive/negative words (Turney, 2002; Esuli and Sebastiani, 2005) to the more complex task of estimating also attitude type and force. As well, a key problem is the fact that evaluative language is often *context-dependent*, and so proper interpretation must consider interactions between a given phrase and its larger textual context.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining generalized association rules. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *Proc. 21st Int. Conf. Very Large Data Bases, VLDB*, pages 407–419. Morgan Kaufmann, 11–15 September.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proc. of ACL Conference on Applied Natural Language Processing*.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. *Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP 2005)*, October.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss analysis. In *Proceedings of CIKM-05, the ACM SIGIR Conference on Information and Knowledge Management*, Bremen, DE.

Michael Gamon, Anthony Aue, Simon Corston-Oliver, and Eric Ringger. 2005. Pulse: Mining customer opinions from free text. In *Proceedings of IDA-05, the 6th International Symposium on Intelligent Data Analysis*, Lecture Notes in Computer Science, Madrid, ES. Springer-Verlag.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

J. R. Martin and P. R. R. White. 2005. *The Language of Evaluation: Appraisal in English*. Palgrave, London. (`http://grammatics.com/appraisal/`).

George A. Miller. 1995. Wordnet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP-04, 9th Conference on Empirical Methods in Natural Language Processing*, Barcelon, ES.

Kamal Nigam and Matthew Hurst. 2004. Towards a robust metric of opinion. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, Standford, US.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. 42nd ACL*, pages 271–278, Barcelona, Spain, July.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT-EMNLP-05, the Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing*, Vancouver, CA.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP*.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings 40th Annual Meeting of the ACL (ACL'02)*, pages 417–424, Philadelphia, Pennsylvania.

Jianyong Wang, Jiawei Han, and Jian Pei. 2003. CLOSET+: searching for the best strategies for mining frequent closed itemsets. In Pedro Domingos, Christos Faloutsos, Ted Senator, Hillol Kargupta, and Lise Getoor, editors, *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pages 236–245, New York, August 24–27. ACM Press.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal taxonomies for sentiment analysis. In *Proceedings of CIKM-05, the ACM SIGIR Conference on Information and Knowledge Management*, Bremen, DE.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational Linguistics*, 30(3).

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, CA.

Whose idea was this, and why does it matter? Attributing scientific work to citations

Advaith Siddharthan & Simone Teufel
Natural Language and Information Processing Group
University of Cambridge Computer Laboratory
{as372,sht25}@cl.cam.ac.uk

Abstract

Scientific papers revolve around citations, and for many discourse level tasks one needs to know whose work is being talked about at any point in the discourse. In this paper, we introduce the *scientific attribution* task, which links different linguistic expressions to citations. We discuss the suitability of different evaluation metrics and evaluate our classification approach to deciding attribution both intrinsically and in an extrinsic evaluation where information about scientific attribution is shown to improve performance on Argumentative Zoning, a rhetorical classification task.

1 Introduction

In the recent past, there has been a focus on information management from scientific literature. In the genetics domain, for instance, information extraction of genes and gene–protein interactions helps geneticists scan large amounts of information (e.g., as explored in the TREC Genomics track (Hersh et al., 2004)). Elsewhere, citation indexes (Garfield, 1979) provide bibliometric data about the frequency with which particular papers are cited. The success of citation indexers such as CiteSeer (Giles et al., 1998) and Google Scholar relies on the robust detection of formal citations in arbitrary text. In bibliographic information retrieval, anchor text, i.e., the context of a citation can be used to characterise (index) the cited paper using terms outside of that paper (Bradshaw, 2003); O'Connor (1982) presents an approach for identifying the area around citations where the text focuses on that citation. And automatic citation classification (Nanba and Okumura, 1999; Teufel et al., 2006) determines the function that a citation plays in the discourse.

For such information access and retrieval purposes, the relevance of a citation within a paper is often crucial. One can estimate how important a citation is by simply counting how often it occurs in the paper. But as Kim and Webber (2006) argue, this ignores many expressions in text which refer to the cited author's work but which are not as easy to recognise as citations. They address the resolution of instances of the third person personal pronoun *"they"* in astronomy papers: it can either refer to a citation or to some entities that are part of research within the paper (e.g., planets or galaxies). Several applications should profit in principle from detecting connections between referring expressions and citations. For instance, in citation function classification, the task is to find out if a citation is described as flawed or as useful. Consider:

> Most computational models of discourse are based primarily on an analysis of the intentions of the speakers **[Cohen and Perrault, 1979][Allen and Perrault, 1980][Grosz and Sidner, 1986]**$^{\text{WEAK}}$. The speaker will form intentions **based on** his goals and then act on these intentions, producing utterances. The hearer will then reconstruct a model of the speaker's intentions upon hearing the utterance. This approach has many strong points, but **does not provide a very satisfactory account** of the adherence to discourse conventions in dialogue.

The three citations above are described as flawed (detectable by *"does not provide a very satisfactory account"*), and thus receive the label Weak. However, in order to detect this, one must first realise that *"this approach"* refers to the three cited papers. A contrasting hypothesis could be that the citations are *used* (thus deserving the label USE; the cue phrase *"based on"* might make us think so (as in the context *"our work is based on"*). This, however, can be ruled out if we know that *"the speaker"* is not referring to some aspect of the current paper.

2 The scientific attribution task

We define an attribution task where possible referents are members of the reference list (i.e., each cited paper), the CURRENT-PAPER, and a back-off category NO-SPECIFIC-PAPER for markables that are not attributable to any specific paper(s). Our markables are as follows: all definite descriptions (e.g., *"the hearer"*, and including demonstrative noun phrases such as *"these intentions"*), all "work" nouns[1], and all pronouns (possessive, personal and demonstrative); c.f., underlined strings in the above example. Our notion of attribution link encompasses two relations:

1. ANAPHORIC: The referents are entire research papers, or the papers' authors
2. SUBPART: The referents are some component of an approach/argument/claim in the research paper

There are two tasks: attributing a linguistic expression to the right paper (including the current paper) – a task we call *scientific attribution* – and deciding whether or not the expression is anaphoric to the entirety of the paper, or just to some subpart of it.

Kim and Webber (2006) solve the problem of distinguishing between these relations for one case. They decide whether the pronoun *"they"* anaphorically refers to the authors of a cited paper, or whether it refers to some entity that is discussed in (a subpart of) a paper (e.g., *"galaxies"*). In this paper, we tackle the other problem of scientific attribution.

We do not distinguish between the two types of links stated above, but only identify which citation(s) a linguistic expression is attributable to. For tasks of interest to us, it is not enough to only consider anaphoric references to entire papers; authors often make statements comparing/using/criticising *aspects* or *subparts* of cited work. We therefore consider a far wider range of markables than Kim and Webber's single pronoun *"they"*.

Our attribution task differs from the traditional anaphora resolution task in that we have a fixed list of possible referents (the reference list items, CURRENT-PAPER or NO-SPECIFIC-PAPER) that are known upfront. Also, we do not form co-reference chains; we attribute a referring expression directly to one or more referents. Ours is therefore a multi-label classification task, where the citations, CURRENT-PAPER and NO-SPECIFIC-PAPER are the labels, and where one or more labels are assigned to each markable.

We evaluate intrinsically by comparing to human-annotated attribution, and extrinsically by showing that automatically acquired knowledge about scientific attribution improves performance on a discourse classification task—*Argumentative Zoning* (Teufel and Moens, 2002), where sentences are labelled as one of {OWN, OTHER, BACKGROUND, TEXTUAL, AIM, BASIS, CONTRAST} according to their role in the author's argument.

We describe our data in §3 and methodology in §4, discuss evaluation metrics in §5, and evaluate intrinsically in §6 and extrinsically in §7.

3 Data

We used data from the CmpLg (Computation and Language archive; 320 conference articles in computational linguistics). The articles are in XML format.

We produced an annotated corpus (10 articles, 4290 data points, i.e., markables) based on written guidelines. The task was found to be quite intuitive by our annotators, and this was reflected in high agreement - Krippendorff's alpha[2] of more than 0.8 (2 annotators, 3 papers, 1429 data points) on the attribution task. The distribution of classes was, as expected, quite skewed: 69% of markables are attributable to

[1]We use a list of around 40 research methodology related nouns from Teufel and Moens (2002), such as e.g., *"study, account, investigation, result"* etc. These are nouns we are particularly interested in.

[2]see description in §5.2

Current-Paper, 7% to no specific paper and 24% to specific references (on average, 1.7 per reference). Details about the annotation process and human agreement figures can be found in Siddharthan and Teufel (2007).

4 Machine Learning Approach

We frame the attribution problem as a classification task: Given a markable (the definite description/pronoun/work noun under consideration), a binary yes/no decision is made for each cited paper, and a binary yes/no decision is made for whether the markable is attributable to the current paper. The list of labels for the markable is compiled by including all the citations for which the machine learner returns yes, and Current-Paper if the learner returns yes. If the list is empty (learner returns no for everything), the label is No-Specific-Paper.

Since the model for whether a markable is attributable to the current work is likely to be different from the model for whether it is attributable to a citation, we trained separate models for the two problems.

4.1 Deciding attribution to a citation

For each data point to be classified (called NP below), we create a machine learning instance for each reference list item by automatically computing the following features from POS-tagged text:

1. Properties of data point (NP) and the closest Citation instance (CIT) of the reference list item:
   (a) Type of NP (Definite Description/Work Noun/Pronoun)
   (b) CIT is a self Citation or not
   (c) CIT is syntactic (in running text) or parenthetical
   (d) Is CIT Hobbs' prediction (searching left–right starting from current sentence and then considering previous sentences, is CIT the first citation or reference to current work found)?
2. Distance measures:
   (a) Dist. between NP and CIT measured in words
   (b) Dist. between NP and CIT measured in sentences
   (c) Dist. between NP and CIT measured in paragraphs
   (d) Is CIT after NP in the discourse (cataphor)?
   (e) Distance between CIT and the closest first person pronoun or *"this paper"* in words
3. Contextual:
   (a) Rank of CIT (how many other reference list items are closer)
   (b) Number of times CIT is cited in the paragraph
   (c) Number of times CIT is cited in the whole paper
   (d) Current Section heading (this feature has 5 values: Introduction, Methods, Results, Conclusions, Unrecognised)
4. Agreement:
   (a) Agreement Number (He/She & single author non-self citation)
   (b) Agreement Person (First & Current/Self Citation, Third and Not-Current)

We have a chicken and egg problem with calculating the distance of a reference to current work in 2(e). Unlike citations, these are not unambiguously marked in the text. We calculate distance from the closest first person pronoun (even though these could possibly refer to a self citation, rather than current work) or the phrase "this paper", which can again refer to other citations but predominantly refers to current work.

4.2 Deciding attribution to current work

We use the same features for the second classifier that makes the decision on whether the data point refers to Current-Paper, with the following changes: Features 1(b,c) are removed as they are meaningless; 1(d) checks Hobbs' prediction for a first person pronoun/"this paper", rather than CIT; in 2(a–d), the distance is measured between the closest first person pronoun/"this paper" and the markable, rather than a citation and the markable; similarly, in 3(b,c) we count instances of first person pronoun/"this paper"; for 2(e), we now calculate the distance of the closest citation instance. In short, the same features are used, but current work and citations are swapped.

5 Evaluation Metrics

We consider two evaluation metrics. The first is the scoring system used for the co-reference task in the Message Understanding Conferences MUC-6 and MUC-7. The second is Krippendorff's α. We briefly discuss both below.

5.1 The MUC-6/MUC-7 Metric

The MUC-6/MUC-7 Co-reference evaluation metric (Vilain et al., 1995) works by comparing co-reference classes across two annotated files. Calling one annotation the "model" and the other the "system", for each co-reference class S in the model, $c(S)$ is the minimal number of co-reference links needed to generate the class (this is one less than the cardinality of the class; $c(S) = |S| - 1$). $m(S)$ is the number of "missing" links in the system annotation relative to the co-reference class as marked up in the model. In other words, this is the minimum number of co-reference links that need to be added to the system annotation to fully generate the co-reference class S in the model. Recall error is then $RE(S) = m(S)/c(S)$ and Recall is $R(S) = 1 - RE = \frac{c(S)-m(S)}{c(S)}$. Recall for the entire file (or set of files) is calculated by summing over all co-reference classes in the model:

$$R = \frac{\sum_i c(S_i) - m(S_i)}{\sum_i c(S_i)}$$

Precision (P) is calculated by swapping the model and system and the f-measure ($F = 2R \times P/(R + P)$) is symmetric with respect to both annotations.

5.2 Krippendorff's Alpha

We follow Passonneau (2004) and Poesio and Artstein (2005) in using Krippendorff (1980)'s α metric to compute agreement between annotations. The advantage of α over the more commonly used κ metric is that α allows for partial agreement when annotators assign multiple labels to the same markable; in this case calculating agreement on a markable requires a more graded agreement calculation than the "1 if sets are identical and 0 otherwise" provided for by κ. Krippendorff's α measures disagreement, and allows for the use of distance metrics to calculate partial disagreement. Following Passonneau, we present results using four distance metrics:

1. (N)ominal: Two sets have distance $N = 0$ if they are identical and $N = 1$ if they are not. α calculated using the nominal distance metric is equivalent to κ.
2. (J)accard: Two sets A and B have distance $J = 1 - |A \cap B|/|A \cup B|$. In other words, the distance between two sets is larger, the smaller their intersection and the larger their union.
3. (D)ice: Two sets A and B have distance $D = 1 - 2 \times |A \cap B|/(|A| + |B|)$. In practice, the Dice distance metric behaves similarly to the Jaccard metric, but tends to be smaller, resulting in slightly higher α.
4. (M)ASI: This is the Jaccard distance J weighted by a monotonicity distance m where, $m = 0$ if two sets are identical; $m = 0.33$ if one is a subset of the other; $m = 0.67$ if the intersection and the two set differences are all non-null; $m = 1$ if the two sets are disjoint. Formally, the MASI metric is $M = m \times J$.

As an example, consider two sets $\{a, b, c\}$ and $\{b, c, d\}$. The distances between these sets are $N = 1$, $J = 1-2/4 = 0.5$, $D = 1-2\times2/(3+3) = 0.33$ and $M = 0.67 \times 0.5 = 0.33$.

Krippendorff's α is defined as $\alpha = 1 - D_o/D_e$, where D_o is the observed disagreement and D_e is the disagreement that is expected by chance:

$$D_o = \frac{1}{c(c-1)} \sum_j \sum_k \sum_{k'} n_{jk} n_{jk'} d_{kk'}$$
$$D_e = \frac{1}{c(c-1)} \sum_k \sum_{k'} n_k n_{k'} d_{kk'}$$

In the above formulae, c is the number of coders, n_{jk} is the number of times item j is classed as category k, n_k is the number of times any item is classed as category k and $d_{kk'}$ is the distance between categories k and k'.

Like κ, Krippendorff's α is 1 when there is perfect agreement, 0 when the observed agreement is only what was expected by chance, negative when observed agreement is less than expected by chance and positive when observed agreement is greater than expected by chance.

6 Intrinsic Evaluation Results

We ran a machine learning experiment using 10-fold cross-validation and the memory-based learner IBk[3] (with k=6), using the WEKA toolkit (Witten and Frank, 2000). The performance is shown in Tables 1 and 2. To position these results we compare them with three baseline lower bounds and the human performance upper bound in Table 3. We use three baselines:

[3]Memory based learning gave better results on this task than other learners (NB, HNB, IBk, J48, cf. § 7.3.

Paper	Items	α-N	α-J	α-D	α-M	%A*
0003055	446	.601	.606	.607	.610	85%
0005006	446	.670	.704	.711	.715	81%
0005015	462	.679	.696	.701	.706	81%
0005025	277	.707	.707	.707	.707	86%
0006011	393	.766	.771	.772	.775	88%
0006038	578	.551	.568	.573	.578	79%
0007035	393	.570	.590	.600	.609	90%
0008026	449	.700	.700	.700	.700	87%
0001001	420	.564	.565	.569	.571	88%
0001020	429	.730	.778	.790	.801	88%
AVG.	429	.654	.669	.673	.677	85%

*% Agreement, the conservative estimate measured using the Nominal metric

Table 1: Agreement with Human Gold Standard

- BASE$_M$ (Major Class): All data points are labelled `CURRENT-WORK`
- BASE$_P$ (Previous): Data points are tagged with the most recent label
- BASE$_H$ (Hobbs' Prediction): Data points are tagged with the label found by Hobbs' (1986) search (Search left to right in each sentence, starting from current sentence, then considering previous sentences)

As Table 3 shows, our machine learning approach performs much better than the baselines on all the agreement metrics, and is indeed closer to human performance than to any of the baselines. The MUC evaluation appears to produce highly inflated results on our task – when there is a small set of co-reference classes and one of these classes contains 70% of data points, it takes only a small number of missing links to correct annotations. This results in unreasonably high values, particularly for the majority class baseline of labelling every data point as CURRENT-PAPER. We believe that the α metrics provide a much more realistic estimate of the difficulty of the task and the relative performances of different approaches.

Table 4 shows the performance of the machine learner for each of the three types of linguistic expressions considered. Pronouns are the easiest to resolve, with on average 90% resolved correctly (an agreement with the human gold standard of $\alpha = .71$). This drops to 85% ($\alpha = .68$) for definite descriptions and demonstratives, and further to 78% ($\alpha = .63$) for remaining work nouns (i.e., those not already in a definite noun phrase).

Paper	No. Classes	Recall	Precision	F
0003055	14	.934	.886	.910
0005006	17	.875	.870	.872
0005015	19	.897	.876	.886
0005025	16	.903	.874	.888
0006011	14	.942	.909	.925
0006038	25	.905	.893	.899
0007035	18	.957	.926	.941
0008026	9	.966	.962	.964
0001001	14	.949	.908	.928
0001020	18	.924	.926	.925
TOTAL	164	.924	.903	.913

Table 2: Evaluation using MUC-6/7 software

Algo	α-N	α-J	α-D	α-M	%Agr*	muc-f
Base$_M$	.002	.001	.001	.001	69%	.934
Base$_P$	-.101	-.083	-.081	-.077	19%	.894
Base$_H$	.387	.397	.399	.407	72%	.910
IBk	**.654**	**.669**	**.673**	**.677**	**85%**	**.913**
Hum**	.806	.808	.808	.809	91%	.965

*% Agreement, the conservative estimate measured using the Nominal metric

**Agreement between two human annotators over a subset of the corpus (3 files, 1429 data points)

Table 3: Comparison with Baselines and Human Performance (Averaged results)

While all the features contributed to the reported results, the most important features (in terms of information gain) for deciding attribution to a citation were the paragraph level citation count 3(b), the distance features 2(a,b,c,d), the rank 3(a) and the Hobbs' prediction 1(d). The most important features for deciding attribution to the current paper were the distance features 2(a,c,e), the rank 3(a) and the Hobbs' prediction 1(d).

7 Extrinsic Evaluation

To demonstrate the use of automatic scientific attribution classification, we studied its utility for one well known discourse annotation task: Argumentative Zoning (Teufel and Moens, 2002). Argumentative Zoning (AZ) is the task of applying one of seven discourse level tags (Figure 1) to each sentence in a scientific paper.

These categories model several aspects of scientific papers: from the distinction of segments by who an idea is attributed to (`Own` – `Other` – `Background`), to the judgement of how the au-

Paper	Pronouns		Definites		Work Nouns	
	α_M	$\%_N$	α_M	$\%_N$	α_M	$\%_N$
0003055	.746	94%	.556	83%	.735	87%
0005006	.846	91%	.703	85%	.700	78%
0005015	.662	83%	.692	79%	.787	86%
0005025	.804	89%	.717	87%	.514	78%
0006011	.824	91%	.807	91%	.615	76%
0006038	.603	90%	.609	81%	.430	66%
0007035	.577	94%	.507	91%	.770	87%
0008026	.678	88%	.726	87%	.551	78%
0011001	.562	97%	.633	87%	.377	81%
0011020	.792	90%	.798	92%	.808	89%
AVG.	.709	90%	.675	85%	.629	78%

Table 4: Results for different markable types

Category	**Description**
`Background`	Generally accepted background knowledge
`Other`	Specific other work
`Own`	Own work: method, results, future work
`Aim`	Specific research goal
`Textual`	Textual section structure
`Contrast`	Contrast, comparison, weakness of other solution
`Basis`	Other work provides basis for own work

Figure 1: AZ Annotation scheme

thors relate to other work (`Contrast` – `Basis`) to the rhetorical status of high-level discourse goals (statement of `Aim`; overview of section structure (`Textual`)). Some of these categories (`Background`, `Other` and `Own`) occur in zones that span many sentences. Other categories typically occur in short zones, often just a single sentence (`Textual`, `Aim`, `Contrast`, `Basis`).

In all work to date, classification of sentences into one of the AZ categories has been performed on the basis of features extracted from within the sentence, and a few contextual features such as section heading and location in document. Scientific attribution links previously unresolved noun phrases or pronouns in the sentence to citations. As this provides the machine learner with more information, AZ results should improve.

7.1 AZ Data

The evaluation corpus used is the one from Teufel and Moens (2002). It consists of 80 conference papers in computational linguistics, containing around 12000 sentences. Each of these is manually tagged as one of {OWN, OTH, BKG, BAS, AIM, CTR, TXT}. The reliability observed is reasonable (Kappa=0.71)).

7.2 Features

Following Teufel and Moens (2002), we used supervised ML using features extracted by shallow processing (POS tagging and pattern matching):

- **Lexical (cue phrase) features** consist of three features: the first models occurrence of about 1700 manually identified scientific cue phrases (such as "in this paper"). The cue phrases are classified into semantic groups. The second models the main verb of the sentence, by lookup in a verb lexicon organised by 13 main clusters of verb types (e.g. "change verbs"), and the third models the likely subject of the sentence, by classifying them either as the authors, or other researchers, or none of the above, using an extensive lexicon of regular expressions.
- **Content word features** model occurrence and density of content words in the sentences, where content words are either defined as non-stoplist words in the subsection heading preceding the sentence, or as words with a high TF*IDF score.
- **Linguistic features** include (complex) tense, voice, and presence of an auxiliary.
- **Citation features** detect properties of formal citations in text, such as the occurrence of authors' names in text, the position of a citation in text, and whether the citation is a self citation (i.e., includes any of the authors of the paper itself).
- **Location features:** Rhetorical roles are expected at certain places in the document, for instance, background sentences are more likely to occur at the beginning of the text, and goal statements often occur after about a fifth of the paper. We model this by splitting the text into ten segments and assigning each sentence to the segment it is located in. We also use the section heading as a contextual feature.

Some categories tend to occur in blocks (e.g., OWN, OTHER, BACKGROUND), and the context in terms of the label of the previous sentence has good predictive value. We model this (the

Learner	kappa		Macro-F	
	No Attrib	With Attrib	No	With
NB	.45	.46	.53	.53
HNB	.42	.45	.51	.53
IBk	.34	.36	.39	.39
J48	.38	.41	.41	.48
Stacking	.45	.48	.51	.53

Table 5: Improvement on AZ from using automatic scientific attribution classification.

so-called **History feature**) by running the classifier twice, and including the prediction for the previous sentence as a feature the second time.

Due to practical considerations, we obtained our linguistic features using the RASP part of speech tagger (Briscoe and Carroll, 1995), when in previous work we used the LT TTT (Grover et al., 2000). We would not expect this to influence results much, however. Another difference is that we use around 1700 additional cue phrases acquired from previous work on another discourse task[4] (Teufel et al., 2006).

In addition to these features, we use four features obtained from the scientific attribution task described in this paper:

Scientific Attribution Features:

- Whether there is any reference to current work in the sentence
- Whether there is any reference to any specific citation in the sentence
- Whether there is any reference in the sentence to work that is in neither the current paper nor any specific citation
- Which of these, if any, is in subject position

Our aim is to explore whether these features obtained from the scientific attribution task influence machine learning performance on AZ.

7.3 AZ results

We ran five different machine learners with and without the four scientific attribution features (c.f., §7.2). Note that our labelled data for the attribution task does not overlap with the 80 papers in the AZ corpus, and all attribution predictions used in features for this AZ experiment

[4]These cues are acquired manually from files that are not part of the AZ evaluation corpus.

	Without Attribution Features						
	Aim	Ctr	Txt	Own	Bkg	Bas	Oth
P	.44	.42	.52	.84	.46	.34	.47
R	.61	.30	.68	.88	.45	.37	.37
F	.52	.35	.59	.86	.46	.35	.42

Correctly Classified Instances	73.0%
Kappa statistic	0.45
Macro-F	0.51

	With Attribution Features						
	Aim	Ctr	Txt	Own	Bkg	Bas	Oth
P	.57	.42	.57	.84	.44	.40	.55
R	.61	.27	.66	.90	.47	.43	.42
F	.59	.33	.61	.87	.46	.41	.47

Correctly Classified Instances	74.7%
Kappa statistic	0.48
Macro-F	0.53

Table 6: Best AZ results using Stacked classifier: with and without Attribution Features.

are obtained entirely from unseen (and indeed unlabelled) data based on the model learnt on 10 papers (c.f., §6). The learners we used (with default WEKA settings) are:

- NB: Naive Bayes learner
- HNB: Hidden Naive Bayes learner
- IBk: Memory based learner
- J48: Decision tree based learner
- STACKING: combining NB and J48 classifiers with the stacking method

As mentioned under *History feature* above, we run each learner twice, the second time including the machine learning prediction for the previous sentence (as we found in Teufel and Moens (2002) for NB, we noticed a slight improvement in performance when using the history feature (between .005 and .01 on both κ and Macro-F for all learners)). We found an improvement from including the four reference features with all the learners, as shown in Table 5.

For a more detailed view of where the improvement comes from, refer to Table 6, which shows precision, recall and f-measure per category for our best learner. The biggest improvements from using attribution features are for the categories OTHER, AIM and BAS. The improvement in OTHER was to be be expected, as this zone is directly related to the attribution classification. The large improvements in AIM and

	Aim	Ctr	Txt	Own	Bkg	Bas	Oth
P	.44	.34	.57	.84	.40	.37	.52
R	.65	.20	.66	.88	.50	.40	.39
F	.52	.26	.61	.86	.44	.38	.44

Correctly Classified Instances	72.5%
Kappa statistic	0.45
Macro-F	0.50

Table 7: Teufel and Moens (2002)'s best AZ results (Naive Bayes Classifier).

Bas is good news, as these are amongst our most informative rhetorical categories for downstream tasks. Our best results of Kappa=0.48 and Macro-F=0.53 are better than the best previously published results on task (Kappa=0.45 and Macro-F=0.50 in Teufel and Moens (2002)). Our results improve on the results of Teufel and Moens (2002) (reproduced in Table 7) – both overall and for each individual category.

8 Conclusions

We have described a new reference task - deciding scientific attribution, and demonstrated high human agreement ($\alpha > 0.8$) on this task. Our machine learning solution using shallow features achieves an agreement of $\alpha_M = 0.68$ with the human gold standard, increasing to $\alpha_M = 0.71$ if only pronouns need to be resolved. We have also demonstrated that information about scientific attribution improves results for a discourse classification task (Argumentative Zoning).

We believe that similar improvements can be achieved on other discourse annotation tasks in the scientific literature domain. In particular, we plan to investigate the use of scientific attribution information for the citation function classification task.

Acknowledgements

This work was funded by the EPSRC project SciBorg (EP/C010035/1, Extracting the Science from Scientific Publications).

References

S. Bradshaw. 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Proc. of ECDL*.

T. Briscoe and J. Carroll. 1995. Developing and evaluating a probabilistic LR parser of part-of-speech and punctuation labels. In *Proc. of IWPT-95*, Prague / Karlovy Vary, Czech Republic.

E. Garfield. 1979. *Citation Indexing: Its Theory and Application in Science, Technology and Humanities*. J. Wiley, New York, NY.

C. L. Giles, K. Bollacker, and S. Lawrence. 1998. Citeseer: An automatic citation indexing system. In *Proc. of the Third ACM Conference on Digital Libraries*.

C. Grover, C. Matheson, A. Mikheev, and M. Moens. 2000. LT TTT - A flexible tokenisation tool. In *Proc. of LREC-00*, Athens, Greece.

W. Hersh, R. Bhuptiraju, L. Ross, P. Johnson, A. Cohen, and D. Kraemer. 2004. Trec 2004 genomics track overview. In *Proc. of TREC*.

J. Hobbs. 1986. Resolving Pronoun References. In *Readings in Natural Language*, Grosz, B., Sparck-Jones, K. and Webber, B. (eds.) Morgan Kaufman.

Y. Kim and B. Webber. 2006. Automatic reference resolution in astronomy articles. In *Proc. of 20th International CODATA Conference*, Beijing, China.

K. Krippendorff. 1980. *Content Analysis: An introduction to its methodology*. Sage Publications, Beverly Hills.

H. Nanba and M. Okumura. 1999. Towards multi-paper summarization using reference information. In *Proc. of IJCAI-99*.

J. O'Connor. 1982. Citing statements: Computer recognition and use to improve retrieval. *Information Processing and Management*, 18(3):125–131.

R. Passonneau. 2004. Computing reliability for coreference annotation. In *Proc. of LREC-04*, Lisbon, Portugal.

M. Poesio and R. Artstein. 2005. Annotating (anaphoric) ambiguity. In *Proc. of the Corpus Linguistics Conference*, Birmingham, UK.

A. Siddharthan and S. Teufel. 2007. Whose idea was this? Deciding attribution in scientific literature. In *Proc. of the 6th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC'07)*, Lagos, Portugal.

S. Teufel and M. Moens. 2002. Summarising scientific articles — experiments with relevance an d rhetorical status. *Computational Linguistics*, 28(4):409–446.

S. Teufel, A. Siddharthan, and D. Tidhar. 2006. Automatic classification of citation function. In *Proc. of EMNLP-06*, Sydney, Australia.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of the 6th Message Understanding Conference*, San Francisco.

I. Witten and E. Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Combining Probability-Based Rankers for Action-Item Detection

Paul N. Bennett
Microsoft Research[*]
One Microsoft Way
Redmond, WA 98052
paul.n.bennett@microsoft.com

Jaime G. Carbonell
Language Technologies Institute, Carnegie Mellon
5000 Forbes Ave
Pittsburgh, PA 15213
jgc@cs.cmu.edu

Abstract

This paper studies methods that automatically detect *action-items* in e-mail, an important category for assisting users in identifying new tasks, tracking ongoing ones, and searching for completed ones. Since action-items consist of a short span of text, classifiers that detect action-items can be built from a document-level or a sentence-level view. Rather than commit to either view, we adapt a context-sensitive metaclassification framework to this problem to combine the *rankings* produced by different algorithms as well as different views. While this framework is known to work well for standard classification, its suitability for fusing rankers has not been studied. In an empirical evaluation, the resulting approach yields improved rankings that are less sensitive to training set variation, and furthermore, the theoretically-motivated reliability indicators we introduce enable the metaclassifier to now be applicable in any problem where the base classifiers are used.

1 Introduction

From business people to the everyday person, e-mail plays an increasingly central role in a modern lifestyle. With this shift, e-mail users desire improved tools to help process, search, and organize the information present in their ever-expanding inboxes. A system that ranks e-mails according to the likelihood of containing "to-do" or *action-items* can alleviate a user's time burden and is the subject of ongoing research throughout the literature.

[*]This work was performed primarily while the first author was supported by Carnegie Mellon University.

From: Henry Hutchins <hhutchins@innovative.company.com>
To: Sara Smith; Joe Johnson; William Woolings
Subject: meeting with prospective customers
Hi All,
I'd like to remind all of you that the group from GRTY will be visiting us next Friday at 4:30 p.m. The schedule is:
+ 9:30 a.m. Informal Breakfast and Discussion in Cafeteria
+ 10:30 a.m. Company Overview
+ 11:00 a.m. Individual Meetings (Continue Over Lunch)
+ 2:00 p.m. Tour of Facilities
+ 3:00 p.m. Sales Pitch
In order to have this go off smoothly, I would like to practice the presentation well in advance. *As a result, I will need each of your parts by Wednesday.*
Keep up the good work!
–Henry

Figure 1: An E-mail with Action-Item (italics added).

In particular, an e-mail user may not always process all e-mails, but even when one does, some emails are likely to be of greater response urgency than others. These messages often contain action-items. Thus, while importance and urgency are not equal to action-item content, an effective action-item detection system can form one prominent subcomponent in a larger prioritization system.

Action-item detection differs from standard text classification in two important ways. First, the user is interested both in detecting whether an email contains action-items and in locating exactly where these action-item requests are contained within the email body. Second, action-item detection attempts to recover the sender's intent — whether she means to elicit response or action on the part of the receiver.

In this paper, we focus on the primary problem of presenting e-mails in a *ranked order* according to their likelihood of containing an action-item. Since action-items typically consist of a short text span — a phrase, sentence, or small passage — supervised input to a learning system can either come at the *document-level* where an e-mail is labeled yes/no as to whether it contains an action-item or at the *sentence-level* where each span that is an action-item is explicitly identified. Then, a corresponding document-level classifier or aggregated predictions from a sentence-level classifier can be used to estimate the overall likelihood for the e-mail.

Rather than commit to either view, we use a combination technique to capture the information each viewpoint has to offer on the current example. The STRIVE approach (Bennett *et al.*, 2005) has been shown to provide robust combinations of heterogeneous models for standard topic classification by capturing areas of high and low reliability via the use of reliability indicators.

However, using STRIVE in order to produce improved rankings has not been previously studied. Furthermore, while they introduce some reliability indicators that are general for text classification problems as well as ones specifically tied to naïve Bayes models, they do not address other classification models. We introduce a series of reliability indicators connected to areas of high/low reliability in kNN, SVMs, and decision trees to allow the combination model to include such factors as the sparseness of training example neighbors around the current example being classified. In addition, we provide a more formal motivation for the role these variables play in the resulting metaclassification model.

Empirical evidence demonstrates that the resulting approach yields a context-sensitive combination model that improves the quality of rankings generated as well as reducing the variance of the ranking quality across training splits.

2 Problem Approach

In contrast to related combination work, we focus on improving *rankings* through the use of a metaclassification framework. In addition, rather than simply focusing on combining models from different classification algorithms, we also examine combining models that have different *views*, in that both the qualitative nature of the labeled data and the application of the learned base models differ. Furthermore, we improve upon work on context-sensitive combination by introducing reliability indicators which model the sensitivity of a classifier's output around the current prediction point. Finally, we focus on the application of these methods to action-item data — a growing area of interest which has been demonstrated to behave differently than more standard text classification problems (*e.g. topic*) in the literature (Bennett and Carbonell, 2005).

2.1 Action-Item Detection

There are three basic problems for action-item detection. (1) *Document detection*: Classify an e-mail as to whether or not it contains an action-item. (2) *Document ranking*: Rank the e-mails such that all e-mail containing action-items occur as high as possible in the ranking. (3) *Sentence detection*: Classify each sentence in an e-mail as to whether or not it is an action-item.

Here we focus on the document ranking problem. Improving the overall ranking not only helps users find e-mails with action-items quicker (Bennett and Carbonell, 2005) but can decrease response times and help ensure that key e-mails are not overlooked.

Since a typical user will eventually process all received mail, we assume that producing a quality ranking will more directly measure the impact on the user than accuracy or F1. Therefore, we focus on ROC curves and area under the curve (AUC) since both reflect the quality of the ranking produced.

2.2 Combining Classifiers with Metaclassifiers

One of the most common approaches to classifier combination is stacking (Wolpert, 1992). In this approach, a metaclassifier observes a past history of classifier predictions to learn how to weight the classifiers according to their demonstrated accuracies and interactions. To build the history, cross-validation over the *training* set is used to obtain predictions from each base classifier. Next, a metalevel representation of the training set is constructed where each example consists of the class label and the predictions of the base classifiers. Finally, a metaclassifier is trained on the metalevel representation to learn a model of how to combine the base classifiers.

However, it might be useful to augment the history with information other than the predicted probabilities. For example, during peer review, reviewers

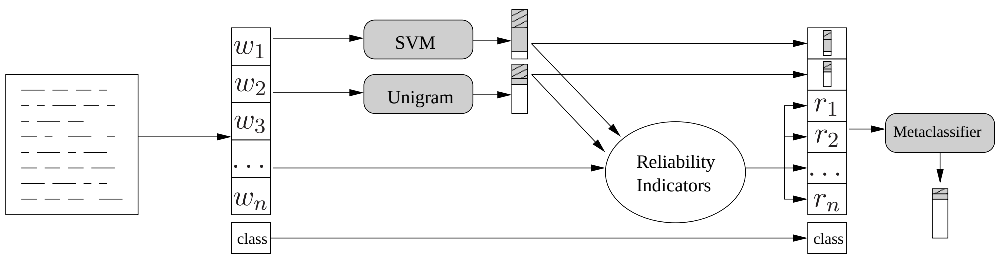


Figure 2: Architecture of *STRIVE*. In *STRIVE*, an additional layer of learning is added where the metaclassifier can use the context established by the reliability indicators and the output of the base classifiers to make an improved decision.

typically provide both a 1-5 acceptance rating and a 1-5 confidence. The first of these is related to an estimate of class membership, $P(\text{“accept''} \mid \text{paper})$, but the second is closer to a measure of expertise or a self-assessment of the reviewer's reliability on an example-by-example basis.

Automatically deriving such self-assessments for classification algorithms is non-trivial. The **St**acked **R**eliability **I**ndicator **V**ariable **E**nsemble framework, or *STRIVE*, demonstrates how to extend stacking by incorporating such self-assessments as a layer of reliability indicators and introduces a candidate set of functions (Bennett *et al.*, 2005).

The *STRIVE* architecture is depicted in Figure 2. From left to right: (1) a bag-of-words representation of the document is extracted and used by the base classifiers to predict class probabilities; (2) reliability indicator functions use the predicted probabilities and the features of the document to characterize whether this document falls within the "expertise" of the classifiers; (3) a metalevel classifier uses the base classifier predictions and the reliability indicators to make a more reliable combined prediction.

From the perspective of improving action-item rankings, we are interested in whether stacking *or* striving can improve the quality of rankings. However, we hypothesize that striving will perform better since it can learn a model that varies the combination rule based on the current example and thus, better capture when a particular classifier at the document-level or sentence-level, bag-of-words or n-gram representation, *etc.* will produce a reliable prediction.

2.3 Formally Motivating Reliability Indicators

While *STRIVE* has been shown to provide robust combination for topic classification, a formal motivation is lacking for the type of reliability indicators that are the most useful in classifier combination.

Assume we restrict our choice of metaclassifier to a linear model. One natural choice is to rank the e-mails according to the estimated posterior probability, $\hat{P}(\text{class} = \text{action item} \mid \mathbf{x})$, but in a linear combination framework it is actually more convenient to work with the estimated log-odds or logit transform which is monotone in the posterior, $\hat{\lambda} = \log \frac{\hat{P}(\text{class=action item}|\mathbf{x})}{1-\hat{P}(\text{class=action item}|\mathbf{x})}$ (Kahn, 2004).

Now, consider applying a metaclassifier to a single base classifier. Given only a classifier's probability estimates, a metaclassifier cannot improve on the estimates if they are well-calibrated (DeGroot and Fienberg, 1986). Thus a metaclassifier applied to a single base classifier corresponds to recalibration (Kahn, 2004).

Assume each of the n base models gives an uncalibrated log-odds estimate $\hat{\lambda}_i$. Then the combination model would have the form $\hat{\lambda}^*(\mathbf{x}) = W_0(\mathbf{x}) + \sum_{i=1}^{n} W_i(\mathbf{x})\hat{\lambda}_i(\mathbf{x})$ where the W_i are example dependent weight functions that the combination model learns. The obvious implication is that our reliability indicators can be informed by the optimal values for the weighting functions.

We can determine the optimal weights in a simplified case with a single base classifier by assuming we are given "true" log-odds values, λ, and a family of distributions $\Delta_{\mathbf{x}}$ such that $\Delta_{\mathbf{x}} = p(\mathbf{z} \mid \mathbf{x})$ encodes what is local to $\mathbf{x}$ by giving the probability of drawing a point $\mathbf{z}$ near to $\mathbf{x}$. We use Δ instead of $\Delta_{\mathbf{x}}$ for notational simplicity. Since Δ encodes the example dependent nature of the weights, we can drop $\mathbf{x}$ from the weight functions. To find weights that minimize the squared difference between the true log-odds and the estimated log-odds in the Δ vicinity of $\mathbf{x}$, we can solve a standard regression problem, $\operatorname{argmin}_{w_0,w_1} E_\Delta \left[\left(w_1 \hat{\lambda} + w_0 - \lambda \right)^2 \right]$. Under the assumption $\text{VAR}_\Delta \left[\hat{\lambda} \right] \neq 0$, this yields:

$$w_0 = E_\Delta[\lambda] - w_1 E_\Delta\left[\hat{\lambda}\right] \quad (1)$$

$$w_1 = \frac{\mathrm{COV}_\Delta\left[\hat{\lambda}, \lambda\right]}{\mathrm{VAR}_\Delta\left[\hat{\lambda}\right]} = \frac{\sigma_\lambda}{\sigma_{\hat{\lambda}}}\rho_{\lambda,\hat{\lambda}} \quad (2)$$

where σ and ρ are the stdev and correlation coefficient under Δ, respectively. The first parameter is a measure of calibration that addresses the question, "How far off on average is the estimated log-odds from the true log-odds in the local context?" The second is a measure of correlation, "How closely does the estimated log-odds vary with the true log-odds?" Note that the second parameter depends on the local sensitivity of the base classifier, $\mathrm{VAR}_\Delta^{1/2}\left[\hat{\lambda}\right] = \sigma_{\hat{\lambda}}$. Although we do not have true log-odds, we *can* introduce local density models to estimate the local sensitivity of the model.

In particular, we introduce a series of reliability indicators by first defining a Δ distribution and either computing $\mathrm{VAR}_\Delta\left[\hat{\lambda}\right]$, $E_\Delta\left[\hat{\lambda}\right]$ or the closely related terms $\mathrm{VAR}_\Delta\left[\hat{\lambda}(\mathbf{z}) - \hat{\lambda}(\mathbf{x})\right]$, $E_\Delta\left[\hat{\lambda}(\mathbf{z}) - \hat{\lambda}(\mathbf{x})\right]$. We use the resulting values for an example as features for a linear metaclassifier. Thus we use a context-dependent bias term but leave the more general model for future work.

2.4 Model-Based Reliability Indicators

As discussed in Section 2.3, we wish to define local distributions in order to compute the local sensitivity and similar terms for the base classification models. To do so, we define local distributions that have the same "flavor" as the base classification model.

First, consider the kNN classifier. Since we are concerned with how the decision function would change as we move locally around the current prediction point, it is natural to consider a set of shifts defined by the k neighbors. In particular, let $\mathbf{d}_i$ denote the document that has been shifted by a factor β_i toward the ith neighbor, *i.e.* $\mathbf{d}_i = \mathbf{d} + \beta_i(\mathbf{n}_i - \mathbf{d})$. We use the largest β_i such that the closest neighbor to the new point is the original document, *i.e.* the boundary of the Voronoi cell (see Figure 3). Clearly, β_i will not exceed 0.5, and we can find it efficiently using a simple bisection algorithm. Now, let Δ be a uniform point-mass distribution over the shifted points and $\hat{\lambda}_{k\mathrm{NN}}$, the output score of the kNN model.

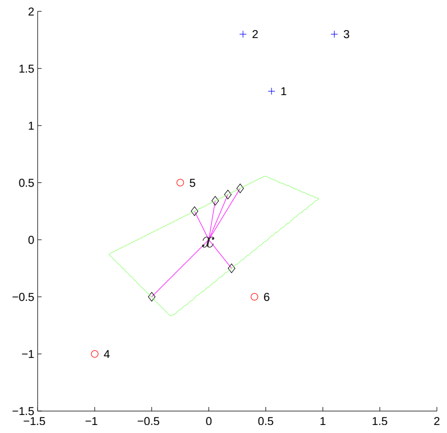


Figure 3: Illustration of the kNN shifts produced for a prediction point x using the numbered points as its neighborhood.

Given this definition of Δ, it is now straightforward to compute the kNN based reliability indicators: $E_\Delta[\hat{\lambda}_{k\mathrm{NN}}(\mathbf{z}) - \hat{\lambda}_{k\mathrm{NN}}(\mathbf{x})]$ and $\mathrm{Var}_\Delta^{1/2}[\hat{\lambda}_{k\mathrm{NN}}(\mathbf{z}) - \hat{\lambda}_{k\mathrm{NN}}(\mathbf{x})]$.

Similarly, we define variables for the SVM classifier by considering a document's locality in terms of nearby support vectors from the set of support vectors, $\mathcal{V}$. To determine β_i, we define it in terms of the closest support vector in $\mathcal{V}$ to $\mathbf{d}$. Let ϵ be half the distance to the nearest point in $\mathcal{V}$, *i.e.* $\epsilon = \frac{1}{2}\min_{\mathbf{v}\in\mathcal{V}} \|\mathbf{v} - \mathbf{d}\|$. Then $\beta_i = \frac{\epsilon}{\|\mathbf{v}_i - \mathbf{d}\|}$.[1] Thus, the shift vectors are all rescaled to have the same length. Now, we must define a probability for the shift. We use a simple exponential based on ϵ and the relative distance from the document to the support vector defining this shift. Let $\mathbf{d}_i \sim \Delta$ where $P_\Delta(\mathbf{d}_i) \propto \exp(-\|\mathbf{v}_i - \mathbf{d}\| + 2\epsilon)$ and $\sum_{i=1}^{V} P_\Delta(\mathbf{d}_i) = 1$.[2]

Given this definition of Δ, we compute the SVM based reliability indicators: $E_\Delta[\hat{\lambda}_{\mathrm{SVM}}(\mathbf{z}) - \hat{\lambda}_{\mathrm{SVM}}(\mathbf{x})]$ and $\mathrm{Var}_\Delta^{1/2}[\hat{\lambda}_{\mathrm{SVM}}(\mathbf{z}) - \hat{\lambda}_{\mathrm{SVM}}(\mathbf{x})]$.

Space prevents us from presenting all the derivations here. However, we also define decision-tree based variables where the locality distribution gives high probability to documents that would land in nearby leaves. For a multinomial naïve Bayes model (NB), we define a distribution of documents identical to the prediction document except having an occurrence of a single feature deleted. For the multivariate Bernoulli naïve Bayes (MBNB) model

[1]We assume that the minimum distance is not zero. If it is zero, then we return zero for all of the variables.

[2]As is standard to handle different document lengths, we take the distance between documents after they have been normalized to the unit sphere.

that models feature presence/absence, we use a distribution over all documents that has one presence/absence bit flipped from the prediction document. It is interesting to note that the variables from the naïve Bayes models can be shown to be equivalent to variables introduced by Bennett *et al.* (2005) — although those were derived in a different fashion by analyzing the weight a single feature carries with respect to the overall prediction.

Furthermore, from this starting point, we go on to define similar variables of possible interest. Including the two for each model described here, we define 10 kNN variables, 5 SVM variables, 2 decision-tree variables, 6 NB model based variables, and 6 MBNB variables. We describe these variables as well as implementation details and computational complexity results in (Bennett, 2006).

3 Experimental Analysis

3.1 Data

Our corpus consists of e-mails obtained from volunteers at an educational institution and covers subjects such as: organizing a research workshop, arranging for job-candidate interviews, publishing proceedings, and talk announcements. After eliminating duplicate e-mails, the corpus contains 744 messages with a total of 6301 automatically segmented sentences. A human panel labeled each phrase or sentence that contained an explicit request for information or action. 416 e-mails have no action-items and 328 e-mails contain action-items. Additional information such as annotator agreement, distribution of message length, *etc.* can be found in (Bennett and Carbonell, 2005). An anonymized corpus is available at http://www.cs.cmu.edu/~pbennett/action-item-dataset.html.

3.2 Feature Representation

We use two types of feature representation: a bag-of-words representation which uses all unigram tokens as the feature pool; and a bag-of-n-grams where n includes all n-grams where $n \leq 4$. For both representations at both the document-level and sentence-level, we used only the top 300 features by the chi-squared statistic.

3.3 Document-Level Classifiers

kNN

We used a s-cut variant of kNN common in text classification (Yang, 1999) and a tfidf-weighting of the terms with a distance-weighted vote of the neighbors to compute the output. k was set to be $2(\lceil \log_2 N \rceil + 1)$ where N is the number of training points. [3] The score used as the uncalibrated log-odds estimate of being an action-item is:

$$\hat{\lambda}_{k\mathrm{NN}}(\mathbf{x}) = \sum_{\mathbf{n} \in k\mathrm{NN}(\mathbf{x}) | c(\mathbf{n}) = {}^{\text{action}}_{\text{item}}} \cos(\mathbf{x}, \mathbf{n}) \quad - \sum_{\mathbf{n} \in k\mathrm{NN}(\mathbf{x}) | c(\mathbf{n}) \neq {}^{\text{action}}_{\text{item}}} \cos(\mathbf{x}, \mathbf{n}).$$

SVM

We used a linear SVM as implemented in the SVM*light* package v6.01 (Joachims, 1999) with a tfidf feature representation and L2-norm. All default settings were used. SVM's margin score, $\sum \alpha_i y_i \, K(\mathbf{x}_i, \mathbf{x}_j)$, has been shown to empirically behave like an uncalibrated log-odds estimate (Platt, 1999).

Decision Trees

For the decision-tree implementation, we used the WinMine toolkit and refer to this as *Dnet* below (Microsoft Corporation, 2001). Dnet builds decision trees using a Bayesian machine learning algorithm (Chickering *et al.*, 1997; Heckerman *et al.*, 2000). The estimated log-odds is computed from a Laplace correction to the empirical probability at a leaf node.

Naïve Bayes

We use a multinomial naïve Bayes (NB) and a multivariate Bernoulli naïve Bayes classifier (MBNB) (McCallum and Nigam, 1998). For these classifiers, we smoothed word and class probabilities using a Bayesian estimate (with the word prior) and a Laplace m-estimate, respectively. Since these are probabilistic, they issue log-odds estimates directly.

3.4 Sentence-Level Classifiers

Each e-mail is automatically segmented into sentences using RASP (Carroll, 2002). Since the corpus has fine grained labels, we can train classifiers to classify a sentence. Each classifier in Section 3.3 is also used to learn a sentence classifier. However, we then must make a document-level prediction.

In order to produce a ranking score, the confidence that the document contains an action-item is:

$$\hat{\lambda}(\mathbf{d}) = \begin{cases} \frac{1}{n(\mathbf{d})} \sum_{s \in \mathbf{d} | \pi(s) = 1} \hat{\lambda}(s), & \exists s \in \mathbf{d} | \pi(s) = 1 \\ \frac{1}{n(\mathbf{d})} \max_{s \in \mathbf{d}} \hat{\lambda}(s) & o.w. \end{cases}$$

[3]This rule is not guaranteed be optimal for a particular value of N but is motivated by theoretical results which show such a rule converges to the optimal classifier as the number of training points increases (Devroye *et al.*, 1996).

where s is a sentence in document $\mathbf{d}$, π is the classifier's 1/0 prediction, $\hat{\lambda}$ is the score the classifier assigns as its confidence that $\pi(s) = 1$, and $n(\mathbf{d})$ is the greater of 1 and the number of (unigram) tokens in the document. In other words, when any sentence is predicted positive, the document score is the length normalized sum of the sentence scores above threshold. When no sentence is predicted positive, the document score is the maximum sentence score normalized by length. The length normalization compensates for the fact that we are more likely to emit a false positive the longer a document is.

3.5 Stacking

To examine the hypothesis that the reliability indicators provide utility beyond the information present in the output of the 20 base classifiers ($2 \text{ representations} * 2 \text{ views} * 5 \text{ classifiers}$), we construct a linear stacking model which uses only the base classifier outputs and no reliability indicators as a baseline. For the implementation, we use SVM*light* with default settings. The inputs to this classifier are normalized to have zero mean and a scaled variance.

3.6 Striving

Since we are constructing base classifiers for both the bag-of-words and bag-of-n-grams representations, this gives 58 reliability indicators from computing the variables in Section 2.4 for the document-level classifiers ($58 = 2 * [6 + 6 + 10 + 5 + 2]$).

Although the model-based indicators are defined for each sentence prediction, to use them at the document-level we must somehow combine the reliability indicators over each sentence. The simplest method is to average each classifier-based indicator across the sentences in the document. We do so and thus obtain another 58 reliability indicators.

Furthermore, our model might benefit from some of the structure a sentence-level classifier offers when combining document predictions. Analogous to the sensitivity of each base model, we can consider such indicators as the mean and standard deviation of the classifier confidences across the sentences within a document. For each sentence-level base classifier, these become two more indicators which we can benefit from when combining document predictions. This introduces 20 more variables ($20 = 2 \text{ representations} * 2 * 5 \text{ classifiers}$).

Finally, we include the 2 basic voting statistic reliability-indicators (*PercentPredictingPositive* and *PercentAgreeWBest*) that Bennett *et al.* (2005) found useful for topic classification. This yields a total of 138 reliability-indicators ($138 = 58 + 20 + 58 + 2$). With the 20 classifier outputs, there are a total of 158 input features for striving to handle.

As with stacking, we use SVM*light* with default settings and normalize the inputs to this classifier to have zero mean and a scaled variance.

3.7 Performance Measures

We wish to improve the rankings of the e-mails in the inbox such that action-item e-mails occur higher in the inbox. Therefore, we use the area under the curve (AUC) of an ROC curve as a measure of ranking performance. AUC is a measure of overall model and ranking quality that has gained wider adoption recently and is equivalent to the Mann-Whitney-Wilcoxon sum of ranks test (Hanley and McNeil, 1982). To put improvement in perspective, we can write our relative reduction in residual area (RRA) as $\frac{1-\text{AUC}}{1-\text{AUC}_{\text{baseline}}}$. We present gains relative to the best AUC performer (bRRA), and relative to perfect dynamic selection performance, (dRRA), which assumes we could accurately *dynamically* choose the best classifier per cross-validation run.

The F1 measure is the harmonic mean of precision and recall and is common throughout text classification (Yang and Liu, 1999). Although we are not concerned with F1 performance here, some users of the system might be interested in improving ranking while having negligible negative effect on F1. Therefore, we examine F1 to ensure that an improvement in ranking will not come at the cost of a statistically significant decrease in F1.

3.8 Experimental Methodology

To evaluate performance of the combination systems, we perform 10-fold cross-validation and compute the average performance. For significance tests, we use a two-tailed t-test (Yang and Liu, 1999) to compare the values obtained during each cross-validation fold with a p-value of 0.05.

We examine two hypotheses: Stacking will outperform all of the base classifiers; Striving will outperform all the base classifiers and stacking.

3.9 Results & Discussion

Table 1 presents the summary of results. The best performer in each column is in bold. If a combination method statistically significantly outperforms all base classifiers, it is underlined.

	F1	AUC	bRRA	dRRA
Document-Level, Bag-of-Words Representation				
Dnet	0.7398	0.8423	1.41	1.78
NB	0.6905	0.7537	2.27	2.91
MBNB	0.6729	0.7745	2.00	2.49
SVM	0.6918	0.8367	1.48	1.87
kNN	0.6695	0.7669	2.17	2.74
Document-Level, Ngram Representation				
Dnet	0.7412	0.8473	1.38	1.77
NB	0.7361	0.8114	1.75	2.23
MBNB	0.7534	0.8537	1.30	1.61
SVM	0.7392	0.8640	1.24	1.59
kNN	0.7021	0.8244	1.62	2.01
Sentence-Level, Bag-of-Words Representation				
Dnet	0.7793	0.8885	1.00	1.27
NB	0.7731	0.8645	1.21	1.50
MBNB	0.7888	0.8699	1.14	1.42
SVM	0.6985	0.8548	1.34	1.70
kNN	0.6328	0.6823	2.98	3.88
Sentence-Level, Ngram Representation				
Dnet	0.7521	0.8723	1.13	1.42
NB	**0.8012**	0.8723	1.15	1.46
MBNB	0.8010	0.8777	1.10	1.38
SVM	0.7842	0.8620	1.23	1.58
kNN	0.6811	0.8078	1.76	2.29
Metaclassifiers				
Stacking	0.7765	0.8996	0.88	1.12
STRIVE	0.7813	**0.9145**	**0.76**	**0.94**

Table 1: Base classifier and combiner performance

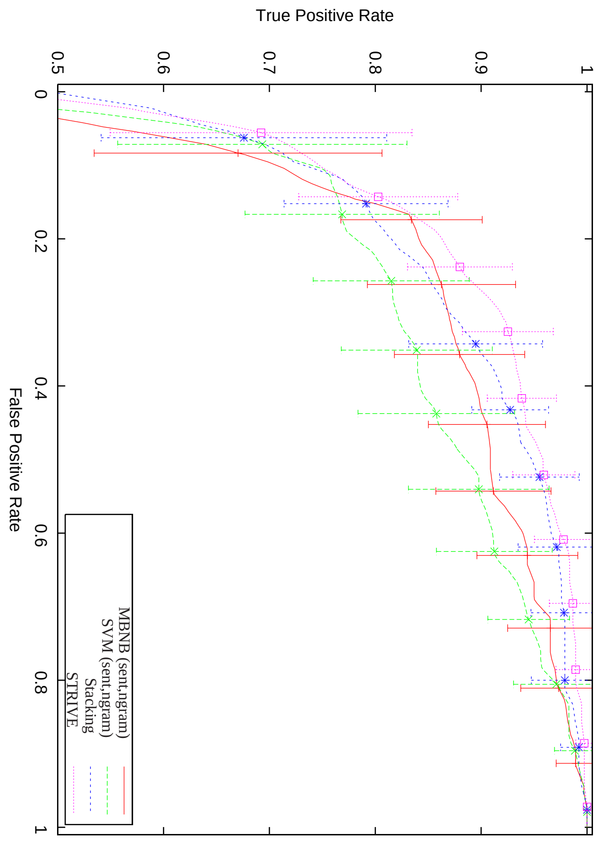


Figure 4: ROC curves (rotated).

Now, we turn to the issue of whether combination improves the ranking of the documents. Examining the results in Table 1, we see that *STRIVE* statistically significantly beats every other classifier according to AUC. Stacking outperforms the base classifiers with respect to AUC but not statistically significantly.

Examining F1, we see that neither combination method outperforms the best base classifier, *NB (sent,ngram)*. If we examine the hypothesis of whether this base classifier significantly outperforms either combination method, the hypothesis is rejected. Thus, STRIVE improves the overall ranking with a negligible effect on F1.

Finally, we compare the ROC curves of striving, stacking, and two of the most competitive base classifiers in Figure 4. We see that striving loses by a slight amount to stacking early in the curve but still beats the base classifiers. Later in the curve, it dominates all the classifiers. If we examine the curves using error bars, we see that the variance of *STRIVE* drops faster than the other classifiers as we move further along the x-axis. Thus, *STRIVE*'s ranking quality varies less with changes to the training set.

4 Related Work

Several researchers have considered text classification tasks similar to action-item detection. Cohen *et al.* (2004) describe an ontology of "speech acts", such as "Propose a Meeting", and attempt to predict when an e-mail contains one of these speech acts. Corston-Oliver *et al.* (2004) consider detecting items in e-mail to "Put on a To-Do List" using a sentence-level classifier. In earlier work (Bennett and Carbonell, 2005), we demonstrated that sentence-level classifiers typically outperform document-level classifiers on this problem and examined the underlying reasons why this was the case. Furthermore, we presented user studies demonstrating that users identify action-items more rapidly when using the system.

In terms of classifier combination, a wide variety of work has been done in the arena. The STRIVE metaclassification approach (Bennett *et al.*, 2005) extended Wolpert's stacking framework (Wolpert, 1992) to use reliability indicators. In recent work, Lee *et al.* (2006) derive variance estimates for naïve Bayes and tree-augmented naïve Bayes and use them in the combination model. Our work complements theirs by laying groundwork for how to compute variance estimates for models such as kNN that have no obvious probabilistic component.

5 Future Work and Conclusion

While there are many interesting directions for future work, the most interesting is to directly integrate the sensitivity and calibration quantities derived into the more general model discussed in Section 2.3.

In this paper, we took an existing approach to context-dependent combination, STRIVE, that used many *ad hoc* reliability indicators and derived a formal motivation for classifier model-based local sensitivity indicators. These new reliability indicators are efficiently computable, and the resulting combination outperformed a vast array of alternative base classifiers for ranking in an action-item detection task. Furthermore, the combination results yielded a more robust performance relative to variation in the training sets. Finally, we demonstrated that the STRIVE method could be successfully applied to ranking.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

References

Paul N. Bennett and Jaime Carbonell. 2005. Feature representation for effective action-item detection. In *SIGIR '05, Beyond Bag-of-Words Workshop*.

Paul N. Bennett, Susan T. Dumais, and Eric Horvitz. 2005. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8:67–100.

Paul N. Bennett. 2006. *Building Reliable Metaclassifiers for Text Learning*. Ph.D. thesis, CMU. CMU-CS-06-121.

John Carroll. 2002. High precision extraction of grammatical relations. In *COLING '02*.

D.M. Chickering, D. Heckerman, and C. Meek. 1997. A Bayesian approach to learning Bayesian networks with local structure. In *UAI '97*.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *EMNLP '04*.

Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. In *Text Summarization Branches Out: Proceedings of the ACL '04 Workshop*.

Morris H. DeGroot and Stephen E. Fienberg. 1986. Comparing probability forecasters: Basic binary concepts and multivariate extensions. In P. Goel and A. Zellner, editors, *Bayesian Inference and Decision Techniques*. Elsevier.

Luc Devroye, László Györfi, and Gábor Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, NY.

James A. Hanley and Barbara J. McNeil. 1982. The meaning and use of the area under a recever operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1:49–75.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In Bernhard Schölkopf, Christopher J. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Joseph M. Kahn. 2004. *Bayesian Aggregation of Probability Forecasts on Categorical Events*. Ph.D. thesis, Stanford University, June.

Chi-Hoon Lee, Russ Greiner, and Shaojun Wang. 2006. Using query-specific variance estimates to combine bayesian classifiers. In *ICML '06*.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI '98, Workshops*. TR WS-98-05.

Microsoft Corporation. 2001. WinMine Toolkit v1.0. http://research.microsoft.com/~dmax/WinMine/ContactInfo.html.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander J. Smola, Peter Bartlett, Bernhard Scholkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *SIGIR '99*.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):67–88.

Multi-document Relationship Fusion via Constraints on Probabilistic Databases

Gideon Mann
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
gideon.mann@gmail.com

Abstract

Previous multi-document relationship extraction and fusion research has focused on single relationships. Shifting the focus to multiple relationships allows for the use of mutual constraints to aid extraction. This paper presents a fusion method which uses a probabilistic database model to pick relationships which violate few constraints. This model allows improved performance on constructing corporate succession timelines from multiple documents with respect to a multi-document fusion baseline.

1 Introduction

Single document information extraction of named entities and relationships has received much attention since the MUC evaluations[1] in the mid-90s (Appelt et al., 1993; Grishman and Sundheim, 1996). Recently, there has been increased interest in the extraction of named entities and relationships from multiple documents, since the redundancy of information across documents has been shown to be a powerful resource for obtaining high quality information even when the extractors have access to little or no training data (Etzioni et al., 2004; Hasegawa et al., 2004). Much of the recent work in multi-document relationship extraction has focused on the extraction of isolated relationships (Agichtein, 2005; Pasca et al., 2006), but often the goal, as in single document tasks like MUC, is to extract a template or a relational database composed of related facts.

With databases containing multiple relationships, the semantics of the database impose constraints on possible database configurations. This paper presents a statistical method which picks relationships which violate few constraints as measured by a probabilistic database model. The constraints are hard constraints, and robust estimates are achieved by accounting for the underlying extraction/fusion uncertainty.

This method is applied to the problem of constructing management succession timelines which have a rich set of semantic constraints. Using constraints on probabilistic databases yields F-Measure improvements of 5 to 18 points on a per-relationship basis over a state-of-the-art multi-document extraction/fusion baseline. The constraints proposed in this paper are used in a context of minimally supervised information extractors and present an alternative to costly manual annotation.

2 Semantic Constraints on Databases

This paper considers management succession databases where each record has three fields: a CEO's name and the start and end years for that person's tenure as CEO (Table 1 Column 1). Each record is represented by three binary logical predicates: *ceo*(c,x), *start*(x,y^1), *end*(x,y^2), where c is a company, x is a CEO's name, and y^1 and y^2 are years.[2]

[1]http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

[2]All of the relationships in this paper are defined to be binary relationships. When extracting relationships of higher ar-

Explicit Relationships	Implicit Relationships	Logical Constraints (a partial list)
ceo(c, x)	*precedes*(x^1,x^2)	*ceo*(c,x^1), *precedes*(x^1,x^2) $\Leftrightarrow$ *ceo*(c,x^2), *end*(x^1,y), *start*(x^2, y)
start(x, y)	*inoffice*(x, y)	*start*(x,y^1), *inoffice*(x,y^2), *end*(x, y^3) $\Rightarrow y^1 \leq y^2 \leq y^3$
end(x, y)	*predates*(x^1, x^2)	*inoffice*(x^1, y^1), *inoffice*(x^2, y^2) , $y^1 < y^2 \Rightarrow$ *predates*(x^1,x^2)
		precedes(x^1,x^2) , *inoffice*(x^1,y^1), *inoffice*(x^2,y^2) $\Rightarrow y^1 \leq y^2$

Table 1: Database semantics can provide 1) a method to augment the explicit relationships in the database with implicit relationships and 2) logical constraints on these explicit and implicit relationships. In the above table, c is a company, x^i is a person, and y^i is a time.

In this setting, database semantics allow for the derivation of other implicit relationships from the database: the immediate predecessor of a given CEO (*precedes*(x^1,x^2)), all predecessors of a given CEO (*predates*(x^1,x^2)) and all years the CEO was in office (*inoffice*(x,y^3)), where x^2 is a CEO's name and t^3 a year (Table 1 Column 2).

These implicit relationships and the original explicit relationships are governed by a series of semantic relations which impose constraints on the permissible database configurations (Table 1 Column 3). For example, it will always be true that a CEO's start date precedes their end date:

$$\forall x : start(x, y^1), end(x, y^2) \Rightarrow y^1 \leq y^2 \,.$$

Multi-document extraction of single relationships exploits redundancy and variety of expression to extract accurate information from across many documents. However, these are not the only benefits of extraction from a large document collection. As well as being rich in redundant information, large document collections also contain a wealth of secondary relationships which are related to the relationship of interest via constraints as described above. These secondary relationships can yield benefits to augment those achieved by redundancy.

3 Multi-Document Database Fusion

There are typically two steps in the extraction of single relationships from multiple documents. In the first step, a relationship extractor goes through the corpus, finds all possible relationships r in all sentences s and gives them a score $p(r|s)$. Next, the relationships are fused across sentences to generate one score ϕ_r for each relationship.

This paper proposes a third step which combines the fusion scores across relationships. This section first presents a probabilistic database model generated from fusion scores and then shows how to use this model for multi-document fusion.

3.1 A Probabilistic Database Model

A relationship r is defined to be a 3-tuple $r^{t,a,b} = r(t, a, b)$, where t is the type of the relationship (e.g. *start*), and a and b are the arguments of the binary relationship.[3]

To construct a probabilistic database for a given corpus, the weights generated in relationship fusion are normalized to provide the conditional probability of a relationship given its type:

$$p(r_1^{t,a,b}|t_1) = \frac{\phi_{r_1^{t,a,b}}}{\sum_{r_i:r_i^t=t} \phi_{r_i^{t,a,b}}},$$

where ϕ_r is the fusion score generated by the extraction/fusion system.[4] By applying a prior over types $p(t)$, a distribution $p(r_1, t_1)$ can be derived. Given strong independence assumptions, the probability of an ordered database configuration $R = r_{1..n}$ of types $t_{1..n}$ is:

$$p(r_{1..n}, t_{1..n}) = \prod_{i=1}^{n} p(r_i, t_i) \,. \tag{1}$$

ity, typically binary relationships are combined (McDonald et al., 2005).

[3]For readibility in future examples, "a" and "b" are replaced by the types of their arguments. For example, for *start* the year in which the CEO starts is referred to as r^{year}.

[4]The following fusion method does not depend on a particular extraction/fusion architecture or training methodology, merely this conditional probability distribution.

As proposed, the model in Equation 1 is faulty since the relationships in a database are not independent. Given a set of database constraints, certain database configurations are **illegal** and should be assigned zero probability. To address this, the model in Equation 1 is augmented with constraints that explicitly set the probability of a database configuration to zero when they are violated.

A database constraint is a logical formula $\eta(r_{1..\pi(\eta)})$, where $\pi(\eta)$ is the arity of the constraint η. For the constraints presented in this paper, all constraints η are modeled with two terms η^α and η^β where:

$$\eta(r_{1..\pi(\eta)}) = \left(\eta^\alpha(r_{1..\pi(\eta)}) \Rightarrow \eta^\beta(r_{1..\pi(\eta)})\right) .$$

For a set of relationships, a constraint **holds** if $\eta(\cdot)$ is true, and the constraint **applies** if $\eta^\alpha(\cdot)$ is true. A constraint $\eta(\cdot)$ can only be **violated** (false) when the constraint applies, since: (false $\Rightarrow X$) = true.

In application to a database, each constraint η is quantified over the database to become a quantified constraint $\eta_{r_{1..n}}$. For example, the constraints that a person's start date must come before their end date is universally quantified over all pairs of relationships in a configuration $R = r_{1..n}$:

$$\begin{aligned}\eta_{r_{1..n}} = \forall r_1, r_2 \in R : \eta(r_1, r_2) = \\ (r_1^t = start, r_2^t = end, r_1^{ceo} = r_2^{ceo}) \\ \Rightarrow (r_1^{year} < r_2^{year}) .\end{aligned}$$

This constraint applies to *start* and *end* relationships whose CEO argument matches and is violated when the years are not in order. If the quantified constraint $\eta_{r_{1..n}}$ is true for a given database configuration $r_{1..n}$ then it **holds**.

To ensure that only legal database configurations are assigned positive probabilities, Equation 1 is augmented with a factor

$$\phi^\eta_{r_{1..n}} = \begin{cases} 1 & \text{if } \eta_{r_{1..n}} \text{holds} \\ 0 & \text{otherwise .} \end{cases}$$

To include a constraint η, the database model in Equation 1 is extended to be:

$$p_\eta(r_{1..n}, t_{1..n}) = \frac{1}{Z}\left(\prod_i p(r_i, t_i)\right)\phi^\eta_{r_{1..n}} ,$$

where Z is the partition function and corresponds to the total probability of all database configurations. A set of constraints $\eta^{1..Q} = \eta^1..\eta^Q$ can be integrated similarly:

$$p_{\eta^{1..Q}}(r_{1..n}, t_{1..n}) = \frac{1}{Z}\left(\prod_i p(r_i, t_i)\right)\prod_q \phi^{\eta^q}_{r_{1..n}} \tag{2}$$

With these added constraints, the probabilistic database model assigns non-zero probability only to databases which don't violate any constraints.

3.2 Constraints on Probabilistic Databases for Relationship Rescoring

Though the constrained probabilistic database model in Equation 2 is theoretically appealing, it would be infeasible to calculate its partition function which requires enumeration of all legal 2^n databases. This section proposes two methods for re-scoring relationships with regards to how likely they are to be present in a legal database configuration using the model proposed above. The first method is a confidence estimate based on how likely it is that η holds for a given relationship r_1:

$$\begin{aligned}\Lambda_\eta(r_1, t_1) &= E_{p(r_{2..n}, t_{2..n})}\left[\phi^\eta_{r_{1..\pi(\eta)}}\right] \\ &= \frac{\sum_{r_{2..\pi(\eta)}}\left(\prod_{i=2}^{\pi(\eta)} p(r_i, t_i)\right)\phi^\eta_{r_{1..\pi(\eta)}}}{\sum_{r_{2..\pi(\eta)}}\left(\prod_{i=2}^{\pi(\eta)} p(r_i, t_i)\right)} \\ &= \frac{\sum_{r_{2..\pi(\eta)}} p_\eta(r_{1..n}, t_{1..n})}{\sum_{r_{2..\pi(\eta)}} p(r_{1..n}, t_{1..n})} ,\end{aligned}$$

where the expectation that the constraint holds is equivalent to the likelihood ratio between the database probability models with and without constraints. In effect, this model measures the expectation that the constraint holds for a finite database "look-ahead" of size $\pi(\eta) - 1$.

With this method, for a constraint to reduce the confidence in a particular relationship by half, half of all configurations would have to violate the constraint.[5] Since inconsistencies are relatively rare, for a given relationship $\Lambda_\eta(r, t) \approx 1$ (i.e. almost all small databases are legal).

[5]Assuming equal probability for all relationships.

To remedy this, another factor ϕ^{α} is defined similarly to ϕ^{η}, except that it takes a value of 1 only if the constraint applies to that database configuration. An applicability probability model is then defined as:

$$p_{\alpha}(r_{1..n}, t_{1..n}) = \frac{1}{Z}\left(\prod_i p(r_i, t_i)\right)\phi^{\alpha}_{r_{1..n}}.$$

The second confidence estimate is based on how likely it is that the constraint is holds in cases where it applies (i.e. is not violated):

$$\begin{aligned}\Lambda_{\eta,\alpha}&(r_1, t_1)\\ &= E_{p_{\alpha}(r_{2..n},t_{2..n})}\left[\phi^{\eta}_{r_{1..\pi(\eta)}}\right]\\ &= \frac{\sum_{r_{2..\pi(\eta)}}\left(\prod_{i=2}^{\pi(\eta)} p(r_i,t_i)\right)\phi^{\alpha}_{r_{1..\pi(\eta)}}\phi^{\eta}_{r_{1..\pi(\eta)}}}{\sum_{r_{2..\pi(\eta)}}\left(\prod_{i=2}^{\pi(\eta)} p(r_i,t_i)\right)\phi^{\alpha}_{r_{1..\pi(\eta)}}}.\end{aligned}$$

When the constraint doesn't apply it cannot be violated, so this confidence estimate ignores those configurations that can't be affected by the constraint.

Recall that $\Lambda_{\eta}(r,t)$ is the likelihood ratio between the probability of configurations in which r holds for constraint η and all configurations. In contrast, $\Lambda_{\eta,\alpha}(r,t)$ is the likelihood ratio between the database configurations where r applies and holds for η and the database configurations where η applies. In the later ratio, for confidence in a particular relationship to be cut in half, only half of the configurations which might actually contain an inconsistency would be required to produce a violation.[6] As a result, $\Lambda_{\eta,\alpha}(r,t)$ gives a much higher penalty to relationships which create inconsistencies than does $\Lambda_{\eta}(r,t)$.

In order to apply multiple constraints, independent database look-aheads are generated for each constraint q:

$$\Lambda_{\eta^{1..Q},\alpha^{1..Q}}(r_1,t_1) = \prod_q \Lambda_{\eta^q,\alpha^q}(r_1,t_1).$$

For a particular relationship type, these confidence scores are calculated and then used to rank the relationships via:

$$\hat{c}_{\eta^{1..Q},\alpha^{1..Q}}(r_1,t_1) = p(r_1,t_1)\prod_q \Lambda_{\eta^q,\alpha^q}(r_1,t_1) \tag{3}$$

Databases with different precision/recall trade offs can be selected by descending the ranked list.[7]

4 Experiments

In order to test the fusion method proposed above, human annotators manually constructed truth data of complete chief executive histories for 18 Fortune-500 companies using online resources. Extraction from these documents is particularly difficult because these data have vast differences in genre and style and are considerably noisy. Furthermore, the task is complicated to start with.[8]

A corpus was created for each company by issuing a Google query for “CEO-of-Company OR Company-CEO”, and collecting the top ranked documents, generating up to 1000 documents per company. The data was then split randomly into training, development and testing sets of 6, 4, and 8 companies.

Training :	Anheuser-Busch, Hewlett-Packard, Lenner, McGraw-Hill, Pfizer, Raytheon
Dev. :	Boeing, Heinz, Staples, Textron
Test :	General Electric, General Motors, Gannett, The Home Depot, IBM, Kroger, Sears, UPS

Ground truth was created from the entire web, but since the corpus for each company is only a small web snapshot, the experimental results are not similar to extraction tasks like MUC and ACE in that the corpus is not guaranteed to contain the information necessary to build the entire database. In particular,

[6]For example, for a *start* relationship and the constraint that a CEO must start before they end, this method would only examine configurations of one *start* and one *end* relationship for the same CEO. The confidence in a particular start date would be halved if half of the proposed end dates for a given CEO occurred before it.

[7]One thing to note is that since all relationships are given confidence estimates separately, this process may result ultimately in a database where constraints are violated. A potential solution, which is not explored here, would be to incrementally add relationships to the database from the ranked list only if their addition doesn't make the database inconsistent.

[8]For example, in certain companies, the title of the chief executive has changed over the years, often going from “President” to “Chief Executive Officer”. To make things more complicated, after the change, the role of “President” may still hang on as a subordinate to the CEO!

1) Only one start or end per person. $\forall r_1, r_2 : \eta(r_1, r_2) = (r_1^{type} = r_2^{type} = (start \cup end), r_1^{ceo} = r_2^{ceo}) \Rightarrow (r_1^{year} = r_2^{year})$
2) Only a CEO's start or end dates belong in the database. $\forall r_1 \exists r_2 : \eta(r_1, r_2) = (r_1^{type} = start \cup end, r_2^{type} = ceo) \Rightarrow (r_1^{ceo} = r_2^{ceo})$
3) Start dates come before end dates. $\forall r_1, r_2 : \eta(r_1, r_2) = (r_1^{type} = start, r_2^{type} = end, r_1^{ceo} = r_2^{ceo}) \Rightarrow (r_1^{year} \leq r_2^{year})$
4) Can't be in the middle of someone else's tenure. $\forall r_1, r_2, r_3 : \eta(r_1, r_2, r_3) = (r_1^{type} = start \cup inoffice, r_2^{type} = end \cup inoffice, r_3^{type} = start \cup inoffice \cup end,$ $r_1^{ceo} = r_2^{ceo} \neq r_3^{ceo}, r_1^{year} < r_2^{year}) \Rightarrow (r_3^{year} \leq r_1^{year} \cup r_3^{year} \geq r_2^{year})$
5) CEO's are only in office after their start. $\forall r_1, r_2 : \eta(r_1, r_2) = (r_1^{type} = start, r_2^{type} = inoffice, r_1^{ceo} = r_2^{ceo}) \Rightarrow (r_1^{year} \leq r_2^{year})$
6) CEO's are only in office before their end. $\forall r_1, r_2 : \eta(r_1, r_2) = (r_1^{type} = inoffice, r_2^{type} = end, r_1^{ceo} = r_2^{ceo}) \Rightarrow (r_1^{year} \leq r_2^{year})$
7) Someone's end is the same as their successor's start. $\forall r_1, r_2, r_3 : \eta(r_1, r_2, r_3) =$ $(r_1^{type} = end, r_2^{type} = start, r_3^{type} = precedes, r_1^{ceo} = r_3^{first}, r_2^{ceo} = r_3^{second}) \Rightarrow (r_1^{year} = r_2^{year})$
8) All of the someone's dates (start, inoffice, end) are before their successors. $\forall r_1, r_2, r_3 : \eta(r_1, r_2, r_3) = (r_1^{type} = start \cup end \cup inoffice, r_2^{type} = start \cup inoffice \cup end, r_3^{type} = precedes,$ $r_1^{ceo} = r_3^{first}, r_2^{ceo} = r_3^{second}) \Rightarrow (r_1^{year} \leq r_2^{year})$
9) Only CEO succession in the database. $\forall r_1 \exists r_1, r_2 : \eta(r_1, r_2, r_3) = (r_1^{type} = precedes, r_2^{type} = r_3^{type} = ceo) \Rightarrow (r_1^{first} = r_2^{ceo}, r_1^{second} = r_3^{ceo})$

Table 2: For a CEO succession database like the one presented in Table 1, the above constraints must hold if the database is consistent.

many CEOs from pre-Internet years were either infrequently mentioned or not mentioned at all in the database.[9] In the following experiments, recall is reported for facts that were retrieved by the extraction system.

4.1 Relationship Extraction and Fusion

A two-class maximum-entropy classifier was trained for each relationship type. Each classifier takes a sentence and two marked entities (e.g. a person and a year)[10] and gives the probability that a relationship between the two entities is supported by the sentence. For each relationship type, one of the elements is designated as the "hook" in order to generate likely negative examples.[11] In training, all entity pairs are collected from the corpus. The pairs whose "hook" element doesn't appear in the database are thrown out. The remaining pairs are then marked by exact match to the database. In testing, the relationship extractor yields the probability $p(r|s)$ of an entity pair relationship r in a particular sentence s.

The features used in the classifier are: unigrams between the given information and the target, distance in words between the given information and the target, and the exact string between the given information and the target (if less that 3 words long).

After extraction from individual sentences, the relationships are fused together such that there is one score for each unique entity pair. In the case of person names, normalization was performed to merge coreferent but lexically distinct names (e.g. "Phil Condit" and "Philip M. Condit").

In the following experiments, the baseline fusion score is:

$$\phi_r = \sum_s p(r|s) \tag{4}$$

4.2 Experimental Results

Given the management succession database proposed in Section 2, Table 2 enumerates a set of quantified constraints. Information extraction and fusion were run separately for each company to create a probabilistic database. In this section, various constraint sets are applied, either individually or jointly, and evaluated in two ways. The first measures per-relationship precision/recall using the model pro-

[9]Another consequence is that assessing the effectiveness of the relationships extraction on a per-extraction basis is difficult. Because there are no training sentences where it is known that the sentence contains the relationship of interest, grading per-extraction results can be deceptive.

[10]The person tagger used is the named-entity tagger from OpenNLP tools and the year tagger simply finds any four digit numbers between 1950 and 2010.

[11]For the CEO relationship, the company was taken to be the hook. For the other relationships the hook was the primary CEO.

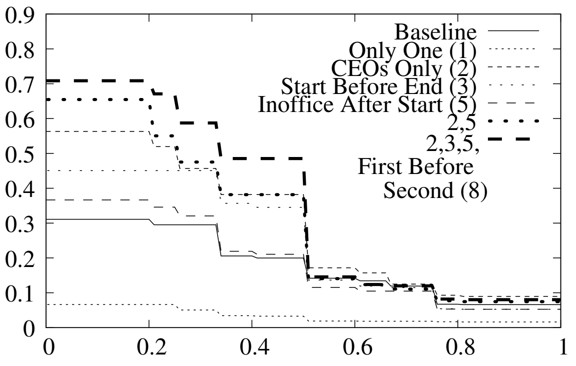


Figure 1: Precision/Recall curve for start(x,t) relationships. The joint constraint "2,3,5,8" is the best performing, even though constraints "3" and "8" (not pictured) alone don't perform well.

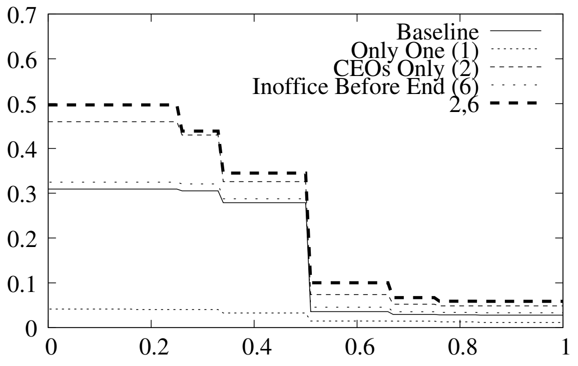


Figure 2: Precision/Recall curve for end(x,t) relationships alone. The joint constraint "2,6" is the best performing.

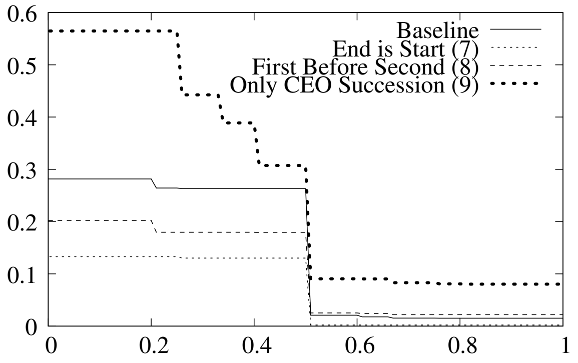


Figure 3: Precision/Recall for precedes(x,y) relationships alone. Though the constraint "First Before Second (8)" helps performance on start(x,t) relationships, the only constraint which aids here is "Only CEOs Succession (9)".

posed and the second looks at the precision/recall of a heterogeneous database with many relationship types. Both evaluations examine the ranked lists of relationships, where the relationships are ranked by rescoring via constraints on probabilistic databases (Equation 3) and compared to the baseline fusion score (Equation 4). The evaluations use two standard metrics, interpolated precision at recall level i (P_{R_i}), and MaxF1:

$$P_{R_i} = \max_{j \geq i} P_{R_j},$$

$$\text{MaxF1} = \max_i \frac{2}{\frac{1}{P_{R_i}} + \frac{1}{R_i}}.$$

Figures 1, 2, and 3 show precision/recall curves for the application of various sets of constraints. Table 3 lists the MaxF1 scores for each of the constraint variants. For *start* and *end*, the majority of constraints are beneficial. For *precedes*, only the constraint that improved performance constraints both people in the relationship to be CEOs. Across all relationships, performance is hurt when using the constraint that there could only be one relationship of each type for a given CEO. The reason behind this is that the confidence estimate based on this constraint favors relationships with few competitors, and those relationships are typically for people who are infrequent in the corpus (and therefore unlikely to be CEOs).

The best-performing constraint sets yield between 5 and 18 points of improvement on Max F1 (Table 3). Surprisingly, the gains from joint constraints are sometimes more than their additive gains. "2,3,5,6,8" is 6 points better for the start relationship than "2,3,5,6", but the gains from "8" alone are negligible.

These performance gains on the individual relationship types also lead to gains when generating an entire database (Figure 4). The highest performing constraint is the "CEOs Only (2)" constraint, which outperforms the joint constraints of the previous section. One reason the joint constraints don't do as well here is that each constraint makes the confidence estimate smaller and smaller. This doesn't have an effect when judging the relationship types individually, but when combining the relationships results, the fused relationships types (*start*, *end*) be-

Constraint Set	Max F1 Start	End	Pre.	DB
∅ (baseline)	31.2	35.8	34.5	37.9
Only One (1)	10.5	7.2	-	38.1
CEOs Only (2) or (9)	43.3	39.4	**39.4**	**42.9**
Start Before End (3)	40.8	32.8	-	40.9
No Overlaps (4)	31.5	35.9	-	36.8
Inoffice After Start (5)	32.5	-	-	38.2
Inoffice Before End (6)	-	36.5	-	37.4
End is Start (7)	7.3	8.0	20.7	39.2
First before Second (8)	31.4	35.6	26.3	38.1
2,5,6	43.3	40.8	-	42.7
2,3,5,6	43.9	43.3	-	42.2
2,3,5,6,8	**49.3**	**43.9**	26.3	40.9

Table 3: Max F1 scores for three relationships **Start**(x,t), **End**(x,t) and **Pre**cedes(x,y)) in isolation and within the context of whole database **DB**. The joint constraints perform best for the explicit relationships in isolation. Using constraints on implicit derived fields (Inoffice and Precedes) provides additional benefit above constraints strictly on explicit database fields (start, end, ceo).

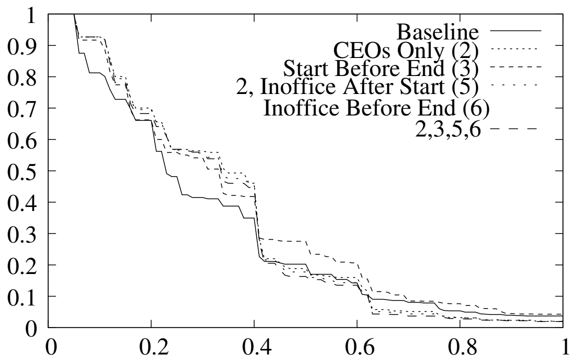


Figure 4: Precision/Recall curve for whole database reconstruction. Performance curves using constraints dominate the baseline.

come artificially lower ranked than the unfused relationship type (*ceo*). The best performing contrained probabilistic database approach beats the baseline by 5 points.

5 Related Work

Techniques for information extraction from minimally supervised data have been explored by Brin (1998), Agichtein and Gravano (2000), and Ravichandran and Hovy (2002). Those techniques propose methods for estimating extractors from example relationships and a corpus which contains instances of those relationships.

Nahm and Mooney (2002) explore techniques for extracting multiple relationships in single document extraction. They learn rules for predicting certain fields given other extracted fields (i.e. a someone who lists Windows as a specialty is likely to know Microsoft Word).

Perhaps the most related work to what is presented here is previous research which uses database information as co-occurrence features for information extraction in a multi-document setting. Mann and Yarowsky (2005) present an incremental approach where co-occurrence with a known relationship is a feature added in training and test. Culotta et al. (2006) introduce a data mining approach where discovered relationships from a database are used as features in extracting new relationships. The database constraints presented in this paper provide a more general framework for jointly conditioning multiple relationships. Additionally, this constraint-based approach can be applied without special training of the extraction/fusion system.

In the context of information fusion of single relationships across multiple documents, Downey et al. (2005) propose a method that models the probabilities of positive and negative extracted classifications. More distantly related, Sutton and McCallum (2004) and Finkel et al. (2005) propose graphical models for combining information about a given entity from multiple mentions.

In the field of question answering, Prager et al. (2004) answer a question about the list of compositions produced by a given subject by looking for related information about the subject's birth and death. Their method treats supporting information as fixed hard constraints on the original questions and are applied in an ad-hoc fashion. This paper proposes a probabilistic method for using constraints in the context of database extraction and applies this method over a larger set of relations.

Richardson and Domingos (2006) propose a method for reasoning about databases and logical constraints using Markov Random Fields. Their model applies reasoning starting from a known database. In this paper the database is built from extraction/fusion of relationships from web pages and contains a significant amount of noise.

6 Conclusion

This paper has presented a probabilistic method for fusing extracted facts in the context of database extraction when there exist logical constraints between the fields in the database. The method estimates the probability than the inclusion of a given relationship will violate database constraints by taking into account the uncertainty of the other extracted relationships. Along with the relationships explicitly listed in the database, constraints are formed over implicit fields directly recoverable from the explicit listed relationships.

The construction of CEO succession timelines using minimally trained extractors from web text is a particularly challenging problem because of noise resulting from the wide variation in genre in the corpora and errors in extraction. The use of constraints on probabilistic databases is effective in resolving many of these errors, leading to improved precision and recall of retrieved facts, with F-measure gains of 5 to 18 points.

The method presented in this paper combines symbolic and statistical approaches to natural language processing. Logical constraints are made more robust by taking into account the uncertainty of the extracted information. An interesting area of future work is the application of data mining to search for appropriate constraints to integrate into this model.

Acknowledgements

This work was supported in part by DoD contract #HM1582-06-1-2013. Any opinions, findings and conclusions or recommendations expressed in this material belong to the author and do not necessarily reflect those of the sponsor.

References

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of ICDL*, pages 85–94.

E. Agichtein. 2005. *Extracting Relations from Large Text Collections*. Ph.D. thesis, Columbia University.

D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. 1993. FASTUS: a finite-state processor for information extraction from real-world text. In *Proceedings of IJCAI*.

S. Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, pages 172–183.

A. Culotta, A. McCallum, and J. Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *HLT-NAACL*, pages 296–303, New York, NY, June.

D. Downey, O. Etzioni, and S. Soderland. 2005. A probabilistic model of redundancy in information extraction. In *IJCAI*.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A-M. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2004. Web-scale information extraction in knowitall. In *WWW*.

J. Finkel, T. Grenager, , and C. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

R. Grishman and B. Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of COLING*.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. Discovering relations amoung named entities from large corpora. In *ACL*.

G. Mann and D. Yarowsky. 2005. Multi-field information extraction and cross-document fusion. In *ACL*.

R. McDonald, F. Pereira, S. Kulick, S. Winters, Y. Jin, and P. White. 2005. Simple algorithms for complex relationship extraction with applications to biomedical ie. In *Proceedings of ACL*.

U. Nahm and R. Mooney. 2002. Text mining with information extraction. In *Proceedings of the AAAI 2220 Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 60–67.

M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extracion challenge. In *AAAI*.

J. Prager, J. Chu-Carroll, and K. Czuba. 2004. Question answering by constraint satisfaction: Qa-by-dossier with constraints. In *Proceedings of ACL*, pages 574–581.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL*, pages 41–47.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July. Presented at ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields.

An Integrated Approach to Measuring Semantic Similarity between Words Using Information available on the Web

Danushka Bollegala
The University of Tokyo
7-3-1, Hongo, Tokyo,
113-8656, Japan
danushka@mi.ci.i.u-tokyo.ac.jp

Yutaka Matsuo
National Institute of Advanced Industrial Science and Technology
1-18-13, Sotokanda, Tokyo,
101-0021, Japan
y.matsuo@aist.go.jp

Mitsuru Ishizuka
The University of Tokyo
7-3-1, Hongo, Tokyo,
113-8656, Japan
ishizuka@i.u-tokyo.ac.jp

Abstract

Measuring semantic similarity between words is vital for various applications in natural language processing, such as language modeling, information retrieval, and document clustering. We propose a method that utilizes the information available on the Web to measure semantic similarity between a pair of words or entities. We integrate page counts for each word in the pair and lexico-syntactic patterns that occur among the top ranking snippets for the *AND* query using support vector machines. Experimental results on Miller-Charles' benchmark data set show that the proposed measure outperforms all the existing web based semantic similarity measures by a wide margin, achieving a correlation coefficient of 0.834. Moreover, the proposed semantic similarity measure significantly improves the accuracy (F-measure of 0.78) in a named entity clustering task, proving the capability of the proposed measure to capture semantic similarity using web content.

1 Introduction

The study of semantic similarity between words has been an integral part of natural language processing and information retrieval for many years. Semantic similarity measures are vital for various applications in natural language processing such as word sense disambiguation (Resnik, 1999), language modeling (Rosenfield, 1996), synonym extraction (Lin, 1998a) and automatic thesaurus extraction (Curran, 2002).

Pre-compiled taxonomies such as WordNet [1] and text corpora have been used in previous work on semantic similarity (Lin, 1998a; Resnik, 1995; Jiang and Conrath, 1998; Lin, 1998b). However, semantic similarity between words change over time as new senses and associations of words are constantly created. One major issue behind taxonomies and corpora oriented approaches is that they might not necessarily capture similarity between proper names such as named entities (e.g., personal names, location names, product names) and the new uses of existing words. For example, *apple* is frequently associated with *computers* on the Web but this sense of apple is not listed in the WordNet. Maintaining an up-to-date taxonomy of all the new words and new usages of existing words is costly if not impossible.

The Web can be regarded as a large-scale, dynamic corpus of text. Regarding the Web as a live corpus has become an active research topic recently. Simple, unsupervised models have shown to perform better when n-gram counts are obtained from the Web rather than from a large corpus (Keller and Lapata, 2003; Lapata and Keller, 2005). Resnik and Smith (2003) extract bilingual sentences from the Web to create parallel corpora for machine translation. Turney (2001) defines a point wise mutual information (PMI-IR) measure using the number of hits returned by a Web search engine to recognize synonyms. Matsuo et. al, (2006b) follows a similar

[1] http://wordnet.princeton.edu/

approach to measure the similarity between words and apply their method in a graph-based word clustering algorithm.

Due to the huge number of documents and the high growth rate of the Web, it is difficult to directly analyze each individual document separately. Search engines provide an efficient interface to this vast information. Page counts and snippets are two useful information sources provided by most Web search engines. Page count of a query is the number of pages that contain the query words [2]. A snippet is a brief window of text extracted by a search engine around the query term in a document. Snippets provide useful information about the immediate context of the query term.

This paper proposes a Web-based semantic similarity metric which combines page counts and snippets using support vector machines. We extract lexico-syntactic patterns from snippets. For example, ***X** is a **Y*** indicates there is a high semantic similarity between ***X*** and ***Y***. Automatically extracted lexico-syntactic patterns have been successfully employed in various term extraction tasks (Hearst, 1992).

Our contributions are summarized as follows:

- We propose a lexico-syntactic patterns-based approach to compute semantic similarity using snippets obtained from a Web search engine.

- We integrate different Web-based similarity scores using WordNet synsets and support vector machines to create a robust semantic similarity measure. The integrated measure outperforms all existing Web-based semantic similarity measures in a benchmark dataset and a named entity clustering task. To the best of our knowledge, this is the first attempt to combine both WordNet synsets and Web content to leverage a robust semantic similarity measure.

2 Previous Work

Given a taxonomy of concepts, a straightforward method for calculating similarity between two words (concepts) is to find the length of the shortest path connecting the two words in the taxonomy (Rada et al., 1989). If a word is polysemous (i.e., having more than one sense) then multiple paths may exist between the two words. In such cases only the shortest path between any two senses of the words is considered for the calculation of similarity. A problem frequently acknowledged with this approach is that it relies on the notion that all links in the taxonomy represent uniform distances.

Resnik (1995) proposes a similarity measure based on information content. He defines the similarity between two concepts C_1 and C_2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C_1 and C_2. Then the similarity between two words are defined as the maximum of the similarity between any concepts that the words belong to. He uses WordNet as the taxonomy and information content is calculated using the Brown corpus.

Li et al., (2003) combines structural semantic information from a lexical taxonomy and information content from a corpus in a non-linear model. They propose a similarity measure that uses shortest path length, depth and local density in a taxonomy. Their experiments using WordNet and the Brown corpus reports a Pearson correlation coefficient of 0.8914 on the Miller and Charles' (1998) benchmark dataset. They do not evaluate their method on similarities between named entities. Recently, some work has been carried out on measuring semantic similarity using web content. Matsuo et al., (2006a) propose the use of Web hits for the extraction of communities on the Web. They measure the association between two personal names using the overlap coefficient, calculated based on the number of Web hits for each individual name and their conjunction.

Sahami et al., (2006) measure semantic similarity between two queries using the snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF weighted term vector. Each vector is L_2 normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors. They do not compare their similarity measure with taxonomy based similarity measures.

Chen et al., (2006) propose a web-based double-

[2]page count may not necessarily be equal to the word frequency because the queried word may appear many times in a page

checking model to compute semantic similarity between words. For two words P and Q, they collect snippets for each word from a web search engine. Then they count the number of occurrences of word P in the snippets for word Q and the number of occurrences of word Q in the snippets for word P. These values are combined non-linearly to compute the similarity between P and Q. This method heavily depends on the search engine's ranking algorithm. Although two words P and Q may be very similar, there is no reason to believe that one can find Q in the snippets for P, or vice versa. This observation is confirmed by the experimental results in their paper which reports 0 similarity scores for many pairs of words in the Miller and Charles (1998) data set.

3 Method

In this section we will describe the various similarity features we use in our model. We utilize page counts and snippets returned by the Google [3] search engine for simple text queries to define various similarity scores.

3.1 Page Counts-based Similarity Scores

For the rest of this paper we use the notation $H(P)$ to denote the page count for the query P in a search engine. Terra and Clarke (2003) compare various similarity scores for measuring similarity between words in a corpus. We modify the traditional Jaccard, overlap (Simpson), Dice and PMI measures for the purpose of measuring similarity using page counts. WebJaccard coefficient between words (or phrases) P and Q, $\mathrm{WebJaccard}(P,Q)$, is defined by,

$$\mathrm{WebJaccard}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c \\ \frac{H(P\cap Q)}{H(P)+H(Q)-H(P\cap Q)} & \text{otherwise} \end{cases} \quad (1)$$

Here, $P \cap Q$ denotes the conjunction query *P AND Q*. Given the scale and noise in the Web, some words might occur arbitrarily, i.e. by random chance, on some pages. Given the scale and noise in web data, it is a possible that two words man order to reduce the adverse effect due to random co-occurrences, we set the WebJaccard coefficient to zero if the page counts for the query $P \cap Q$ is less than a threshold c. [4]

Likewise, we define WebOverlap coefficient, $\mathrm{WebOverlap}(P,Q)$, as,

$$\mathrm{WebOverlap}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c \\ \frac{H(P\cap Q)}{\min(H(P),H(Q))} & \text{otherwise} \end{cases} . \quad (2)$$

We define *WebDice* as a variant of Dice coefficient. $\mathrm{WebDice}(P,Q)$ is defined as,

$$\mathrm{WebDice}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c \\ \frac{2H(P\cap Q)}{H(P)+H(Q)} & \text{otherwise} \end{cases} . \quad (3)$$

We define *WebPMI* as a variant form of PMI using page counts by,

$$\mathrm{WebPMI}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c \\ \log_2\left(\frac{\frac{H(P\cap Q)}{N}}{\frac{H(P)}{N}\frac{H(Q)}{N}}\right) & \text{otherwise} \end{cases} \quad (4)$$

Here, N is the number of documents indexed by the search engine. Probabilities in Formula 4 are estimated according to the maximum likelihood principle. In order to accurately calculate PMI using Formula 4, we must know N, the number of documents indexed by the search engine. Although estimating the number of documents indexed by a search engine (Bar-Yossef and Gurevich, 2006) is an interesting task itself, it is beyond the scope of this work. In this work, we set $N = 10^{10}$ according to the number of indexed pages reported by Google.

3.2 Snippets-based Synonymous Word Patterns

Page counts-based similarity measures do not consider the relative distance between P and Q in a page or the length of the page. Although P and Q occur in a page they might not be related at all. Therefore, page counts-based similarity measures are prone to noise and are not reliable when $H(P\cap Q)$ is low. On the other hand snippets capture the local context of query words. We propose lexico-syntactic patterns extracted from snippets as a solution to the problems with page counts-based similarity measures.

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

[4]we set $c = 5$ in our experiments

To illustrate our pattern extraction algorithm consider the following snippet from Google for the query ***jaguar*** *AND* ***cat***.

*"The **Jaguar** is the largest **cat** in Western Hemisphere and can subdue a larger prey than can the puma"*

Here, the phrase *is the largest* indicates a hypernymic relationship between Jaguar and the cat. Phrases such as *also known as, is a, part of, is an example of* all indicate various of semantic relations. Such indicative phrases have been successfully applied in various tasks such as synonym extraction, hyponym extraction (Hearst, 1992) and fact extraction (Pasca et al., 2006).

We describe our pattern extraction algorithm in three steps.

Step 1

We replace the two query terms in a snippet by two wildcards *X* and *Y*. We extract all word n-grams that contain both *X* and *Y*. In our experiments we extracted n-grams for $n = 2$ to 5. For example, from the previous snippet we extract the pattern, ***X*** *is the largest* ***X***. In order to leverage the pattern extraction process, we randomly select 5000 pairs of synonymous nouns from WordNet synsets. We ignore the nouns which do not have synonyms in the WordNet. For nouns with more than one sense, we select synonyms from its dominant sense. For each pair of synonyms (P, Q), we query Google for "P" AND "Q" and download the snippets. Let us call this collection of snippets as the *positive corpus*. We apply the above mentioned n-gram based pattern extraction procedure and count the frequency of each valid pattern in the positive corpus.

Step 2

Pattern extraction algorithm described in step 1 yields $4,562,471$ unique patterns. $80\%$of these patterns occur less than 10 times in the positive corpus. It is impossible to learn with such a large number of sparse patterns. Moreover, some patterns might occur purely randomly in a snippet and are not good indicators of semantic similarity. To measure the reliability of a pattern as an indicator of semantic similarity we employ the following procedure. We create a set of non-synonymous word-pairs by randomly shuffling the words in our data set of synonymous word-pairs. We check each pair of words in this newly created data set against WordNet and confirm that they do not belong to any of the synsets in the WordNet. From this procedure we created 5000 non-synonymous pairs of words. For each non-synonymous word-pair, we query Google for the conjunction of its words and download snippets. Let us call this collection of snippets as the *negative corpus*. For each pattern generated in step 1, we count its frequency in the negative corpus.

Table 1: Contingency table

	v	other than v	All
Freq. in positive corpus	p_v	$P - p_v$	P
Freq. in negative corpus	n_v	$N - n_v$	N

Step 3

We create a contingency table as shown in Table 1 for each pattern v extracted in step 1 using its frequency p_v in positive corpus and n_v in negative corpus. In Table 1, P denotes the total frequency of all patterns in the positive corpus and N denotes that in the negative corpus.

Using the information in Table 1, we calculate χ^2 (Manning and Schütze, 2002) value for each pattern as,

$$\chi^2 = \frac{(P+N)(p_v(N-n_v) - n_v(P-p_v))^2}{PN(p_v+n_v)(P+N-p_v-n_v)}. \tag{5}$$

We selected the top ranking 200 patterns experimentally as described in section 4.2 according to their χ^2 values. Some of the selected patterns are shown in Table 2.

3.3 Training

For each pair of synonymous and non-synonymous words in our datasets, we count the frequency of occurrence of the patterns selected in Step 3. We normalize the frequency count of each pattern by dividing from the total frequency of all patterns. Moreover, we compute the page counts-based features as given by formulae (1-4). Using the 200 pattern features and the 4 page counts-based features we create 204 dimensional feature vectors for each training instance in our synonymous and non-synonymous datasets. We train a two class support vector machine (SVM) (Vapnik, 1998), where class

$+1$ represents synonymous word-pairs and class -1 represents non-synonymous word-pairs. Finally, SVM outputs are converted to posterior probabilities (Platt, 2000). We consider the posterior probability of a given pair of words belonging to class $+1$ as the semantic similarity between the two words.

4 Experiments

To evaluate the performance of the proposed semantic similarity measure, we conduct two sets of experiments. Firstly, we compare the similarity scores produced by the proposed measure against the Miller-Charles' benchmark dataset. We analyze the performance of the proposed measure with the number of snippets and the size of the training data set. Secondly, we apply the proposed measure in a real-world named entity clustering task and measure its performance.

4.1 The Benchmark Dataset

We evaluated the proposed method against Miller-Charles (1998) dataset, a dataset of 30 [5] word-pairs rated by a group of 38 human subjects. Word-pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy). Miller-Charles' dataset is a subset of Rubenstein-Goodenough's (1965) original dataset of 65 word-pairs. Although Miller-Charles' experiment was carried out 25 years later than Rubenstein-Goodenough's, two sets of ratings are highly correlated (Pearson correlation coefficient=0.97). Therefore, Miller-Charles ratings can be considered as a reliable benchmark for evaluating semantic similarity measures.

4.2 Pattern Selection

We trained a linear kernel SVM with top N pattern features (ranked according to their χ^2 values) and calculated the Pearson correlation coefficient against the Miller-Charles' benchmark dataset. Experimental results are shown in Figure 1. From Figure 1 we select $N = 200$, where correlation maximizes. Features with the highest linear kernel weights are shown in Table 2 alongside with their χ^2 values. The weight of a feature in the linear kernel can be considered as a rough estimate of the influence it has on the

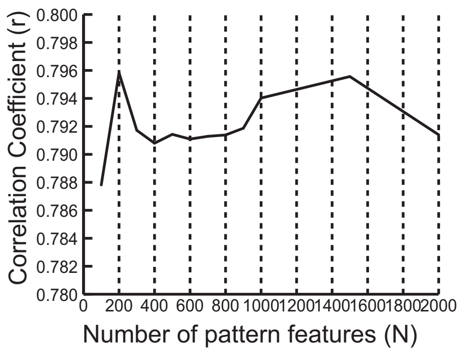


Figure 1: Correlation vs No of pattern features

Table 2: Features with the highest SVM linear kernel weights

feature	χ^2	SVM weight
WebDice	N/A	8.19
X/Y	33459	7.53
X, Y :	4089	6.00
X or Y	3574	5.83
X Y for	1089	4.49
X . the Y	1784	2.99
with X (Y	1819	2.85
X=Y	2215	2.74
X and Y are	1343	2.67
X of Y	2472	2.56

final SVM output. WebDice has the highest linear kernel weight followed by a series of patterns-based features. WebOverlap (rank=18, weight=2.45), WebJaccard (rank=66, weight=0.618) and WebPMI (rank=138, weight=0.0001) are not shown in Table 2 due to space limitations. It is noteworthy that the pattern features in Table 2 agree with the intuition. Lexical patterns (e.g., *X or Y, X and Y are, X of Y*) as well as syntactic patterns (e.g., bracketing, comma usage) are extracted by our method.

4.3 Semantic Similarity

We score the word-pairs in Miller-Charles dataset using the page counts-based similarity measures, previous work on web-based semantic similarity measures (Sahami (2006), Chen (2006)) and the proposed method (SVM). Results are shown in Table 4.3. All figures except for the Miller-Charles ratings are normalized into $[0, 1]$ range for the ease of comparison [6]. Proposed method (SVM) re-

[5] Due to the omission of two word-pairs in earlier versions of WordNet most researchers had used only 28 pairs for evaluations

[6] Pearson correlation coefficient is invariant against a linear transformation

Table 3: Semantic Similarity of Human Ratings and baselines on Miller-Charles dataset

Word Pair	Miller-Charles	Web Jaccard	Web Dice	Web Overlap	Web PMI	Sahami (2006)	Chen (CODC) (2006)	Proposed (SVM)
cord-smile	0.13	0.102	0.108	0.036	0.207	0.090	0	0
rooster-voyage	0.08	0.011	0.012	0.021	0.228	0.197	0	0.017
noon-string	0.08	0.126	0.133	0.060	0.101	0.082	0	0.018
glass-magician	0.11	0.117	0.124	0.408	0.598	0.143	0	0.180
monk-slave	0.55	0.181	0.191	0.067	0.610	0.095	0	0.375
coast-forest	0.42	0.862	0.870	0.310	0.417	0.248	0	0.405
monk-oracle	1.1	0.016	0.017	0.023	0	0.045	0	0.328
lad-wizard	0.42	0.072	0.077	0.070	0.426	0.149	0	0.220
forest-graveyard	0.84	0.068	0.072	0.246	0.494	0	0	0.547
food-rooster	0.89	0.012	0.013	0.425	0.207	0.075	0	0.060
coast-hill	0.87	0.963	0.965	0.279	0.350	0.293	0	0.874
car-journey	1.16	0.444	0.460	0.378	0.204	0.189	0.290	0.286
crane-implement	1.68	0.071	0.076	0.119	0.193	0.152	0	0.133
brother-lad	1.66	0.189	0.199	0.369	0.644	0.236	0.379	0.344
bird-crane	2.97	0.235	0.247	0.226	0.515	0.223	0	0.879
bird-cock	3.05	0.153	0.162	0.162	0.428	0.058	0.502	0.593
food-fruit	3.08	0.753	0.765	1	0.448	0.181	0.338	0.998
brother-monk	2.82	0.261	0.274	0.340	0.622	0.267	0.547	0.377
asylum-madhouse	3.61	0.024	0.025	0.102	0.813	0.212	0	0.773
furnace-stove	3.11	0.401	0.417	0.118	1	0.310	0.928	0.889
magician-wizard	3.5	0.295	0.309	0.383	0.863	0.233	0.671	1
journey-voyage	3.84	0.415	0.431	0.182	0.467	0.524	0.417	0.996
coast-shore	3.7	0.786	0.796	0.521	0.561	0.381	0.518	0.945
implement-tool	2.95	1	1	0.517	0.296	0.419	0.419	0.684
boy-lad	3.76	0.186	0.196	0.601	0.631	0.471	0	0.974
automobile-car	3.92	0.654	0.668	0.834	0.427	1	0.686	0.980
midday-noon	3.42	0.106	0.112	0.135	0.586	0.289	0.856	0.819
gem-jewel	3.84	0.295	0.309	0.094	0.687	0.211	1	0.686
Correlation	1	0.259	0.267	0.382	0.548	0.579	0.693	0.834

ports the highest correlation of 0.8129 in our experiments. Our implementation of Co-occurrence Double Checking (CODC) measure (Chen et al., 2006) reports the second best correlation of 0.6936. However, CODC measure reports zero similarity for many word-pairs. This is because for a word-pair (P, Q), we might not necessarily find Q among the top snippets for P (and vice versa). CODC measure returns zero under these conditions. Sahami et al. (2006) is ranked third with a correlation of 0.5797. Among the four page counts based measures WebPMI reports the highest correlation ($r = 0.5489$). Overall, the results in Table 4.3 suggest that snippet-based measures are more accurate than page counts-based measures in capturing semantic similarity. This is evident for word-pairs where at least one of the words is a polysemous word (e.g., pairs that include *cock*, *brother*). Page counts-based measures do not consider the context in which the words appear in a page, thus cannot disambiguate the multiple senses.

Table 4: Comparison with taxonomy based methods

Method	correlation
Human replication	0.901
Resnik (1995)	0.745
Lin (1998)	0.822
Li et al (2003)	0.891
Edge-counting	0.664
Information content	0.745
Jiang & Conrath (1998)	0.848
proposed (SVM)	0.834

As summarized in Table 4.3, proposed method is comparable with the WordNet based methods. In fact, the proposed method outperforms simple WordNet based approaches such as Edge-Counting and Information Content measures. However, considering the high correlation between human subjects (0.9), there is still room for improvement.

Figure 2 illustrates the effect of the number of snippets on the performance of the proposed

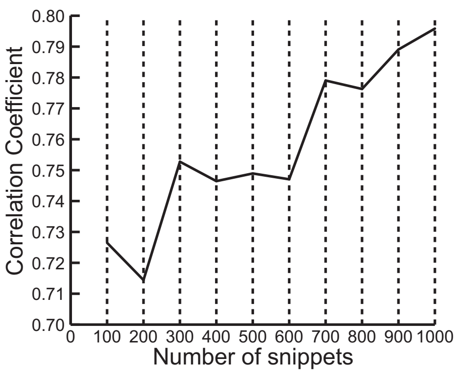


Figure 2: Correlation vs No of snippets

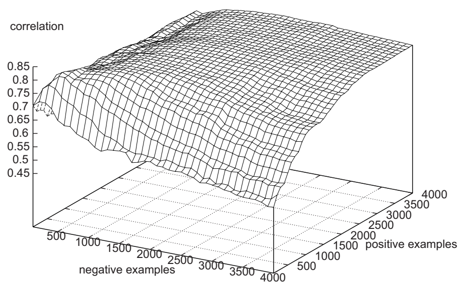


Figure 3: Correlation vs No of positive and negative training instances

method. Correlation coefficient steadily improves with the number of snippets used for extracting patterns. When few snippets are processed only a few patterns are found, thus the feature vector becomes sparse, resulting in poor performance. Figure 3 depicts the correlation with human ratings for various combinations of positive and negative training instances. Maximum correlation coefficient of 0.834 is achieved with 1900 positive training examples and 2400 negative training examples. Moreover, Figure 3 reveals that correlation does not improve beyond 2500 positive and negative training examples. Therefore, we can conclude that 2500 examples are sufficient to leverage the proposed semantic similarity measure.

4.4 Named Entity Clustering

Measuring semantic similarity between named entities is vital in many applications such as query expansion (Sahami and Heilman, 2006) and community mining (Matsuo et al., 2006a). Since most named entities are not covered by WordNet, similarity measures based on WordNet alone cannot be used in such tasks. Unlike common English words, named entities are constantly being created. Manually maintaining an up-to-date taxonomy of named entities is costly, if not impossible. The proposed semantic similarity measure is appealing as it does not require pre-compiled taxonomies. In order to evaluate the performance of the proposed measure in capturing the semantic similarity between named entities, we set up a named entity clustering task. We selected 50 person names from 5 categories : tennis players, golfers, actors, politicians and scientists, (10 names from each category) from the *dmoz* directory [7]. For each pair of names in our dataset, we measure the association between the two names using the proposed method and baselines. We use group-average agglomerative hierarchical clustering to cluster the names in our dataset into five clusters. We employed the B-CUBED metric (Bagga and Baldwin, 1998) to evaluate the clustering results. As summarized in Table 5 the proposed method outperforms all the baselines with a statistically significant ($p \leq 0.01$ Tukey HSD) F score of 0.7897.

Table 5: Performance of named entity clustering

Method	Precision	Recall	F Measure
WebJaccard	0.5926	0.712	0.6147
WebOverlap	0.5976	0.68	0.5965
WebDice	0.5895	0.716	0.6179
WebPMI	0.2649	0.428	0.2916
Sahami (2006)	0.6384	0.668	0.6426
Chen (2006)	0.4763	0.624	0.4984
Proposed	0.7958	0.804	0.7897

5 Conclusion

We propose an SVM-based approach to combine page counts and lexico-syntactic patterns extracted from snippets to leverage a robust web-based semantic similarity measure. The proposed similarity measure outperforms existing web-based similarity measures and competes with models trained on WordNet. It requires just 2500 synonymous word-pairs, automatically extracted from WordNet synsets, for training. Moreover, the proposed method proves useful in a named entity clustering task. In future, we intend to apply the proposed method to automatically extract synonyms from the web.

[7]http://dmoz.org

References

A. Bagga and B. Baldwin. 1998. Entity-based cross document coreferencing using the vector space model. In *Proc. of 36th COLING-ACL*, pages 79–85.

Z. Bar-Yossef and M. Gurevich. 2006. Random sampling from a search engine's index. In *Proceedings of 15th International World Wide Web Conference*.

H. Chen, M. Lin, and Y. Wei. 2006. Novel association measures using web search with double checking. In *Proc. of the COLING/ACL 2006*, pages 1009–1016.

J. Curran. 2002. Ensemble menthods for automatic thesaurus extraction. In *Proc. of EMNLP*.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of 14th COLING*, pages 539–545.

J.J. Jiang and D.W. Conrath. 1998. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the International Conference on Research in Computational Linguistics ROCLING X*.

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

M. Lapata and F. Keller. 2005. Web-based models ofr natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

D. Lin. 1998a. Automatic retreival and clustering of similar words. In *Proc. of the 17th COLING*, pages 768–774.

D. Lin. 1998b. An information-theoretic definition of similarity. In *Proc. of the 15th ICML*, pages 296–304.

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

Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. 2006a. Polyphonet: An advanced social network extraction system. In *Proc. of 15th International World Wide Web Conference*.

Y. Matsuo, T. Sakaki, K. Uchiyama, and M. Ishizuka. 2006b. Graph-based word clustering using web search engine. In *Proc. of EMNLP 2006*.

G. Miller and W. Charles. 1998. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proc. of AAAI-2006*.

J. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74.

R. Rada, H. Mili, E. Bichnell, and M. Blettner. 1989. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):17–30.

P. Resnik and N. A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

P. Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of 14th International Joint Conference on Aritificial Intelligence*.

P. Resnik. 1999. Semantic similarity in a taxonomy: An information based measure and its application to problems of ambiguity in natural language. *Journal of Aritificial Intelligence Research*, 11:95–130.

R. Rosenfield. 1996. A maximum entropy approach to adaptive statistical modelling. *Computer Speech and Language*, 10:187–228.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8:627–633.

M. Sahami and T. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of 15th International World Wide Web Conference*.

E. Terra and C.L.A. Clarke. 2003. Frequency estimates for statistical word similarity measures. In *Proc. of the NAACL/HLT*, pages 165–172.

P. D. Turney. 2001. Minning the web for synonyms: Pmi-ir versus lsa on toefl. In *Proc. of ECML-2001*, pages 491–502.

V. Vapnik. 1998. *Statistical Learning Theory*. Wiley, Chichester, GB.

D. McLean Y. Li, Zuhair A. Bandar. 2003. An approch for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882.

An Information Retrieval Approach to Sense Ranking

Mirella Lapata and **Frank Keller**
School of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, UK
{mlap,keller}@inf.ed.ac.uk

Abstract

In word sense disambiguation, choosing the most frequent sense for an ambiguous word is a powerful heuristic. However, its usefulness is restricted by the availability of sense-annotated data. In this paper, we propose an information retrieval-based method for sense ranking that does not require annotated data. The method queries an information retrieval engine to estimate the degree of association between a word and its sense descriptions. Experiments on the Senseval test materials yield state-of-the-art performance. We also show that the estimated sense frequencies correlate reliably with native speakers' intuitions.

1 Introduction

Word sense disambiguation (WSD), the ability to identify the intended meanings (senses) of words in context, is crucial for accomplishing many NLP tasks that require semantic processing. Examples include paraphrase acquisition, discourse parsing, or metonymy resolution. Applications such as machine translation (Vickrey et al., 2005) and information retrieval (Stokoe, 2005) have also been shown to benefit from WSD.

Given the importance of WSD for basic NLP tasks and multilingual applications, much work has focused on the computational treatment of sense ambiguity, primarily using data-driven methods. Most accurate WSD systems to date are supervised and rely on the availability of training data (see Yarowsky and Florian 2002; Mihalcea and Edmonds 2004 and the references therein). Although supervised methods typically achieve better performance than unsupervised alternatives, their applicability is limited to those words for which sense labeled data exists, and their accuracy is strongly correlated with the amount of labeled data available. Furthermore, current supervised approaches rarely outperform the simple heuristic of choosing the most common or dominant sense in the training data (henceforth "the first sense heuristic"), despite taking local context into account. One reason for this is the highly skewed distribution of word senses (McCarthy et al., 2004a). A large number of frequent content words is often associated with only one dominant sense.

Obtaining the first sense via annotation is obviously costly and time consuming. Sense annotated corpora are not readily available for different languages or indeed sense inventories. Moreover, a word's dominant sense will vary across domains and text genres (the word *court* in legal documents will most likely mean *tribunal* rather than *yard*). It is therefore not surprising that recent work (McCarthy et al., 2004a; Mohammad and Hirst, 2006; Brody et al., 2006) attempts to alleviate the annotation bottleneck by inferring the first sense automatically from raw text. Automatically acquired first senses will undoubtedly be noisy when compared to human annotations. Nevertheless, they can be usefully employed in two important tasks: (a) to create preliminary annotations, thus supporting the "annotate automatically, correct manually" methodology used to provide high volume annotation in the Penn Treebank project; and (b) in combination with supervised WSD methods that take context into account; for instance, such methods could default to the dominant sense for unseen words or words with uninformative contexts.

This paper focuses on a knowledge-lean sense ranking method that exploits a sense inventory like WordNet and corpus data to automatically induce dominant senses. The proposed method infers the associations between words and sense descriptions automatically by querying an IR engine whose index terms have been compiled from the corpus of interest. The approach is inexpensive, language-independent, requires minimal supervision, and uses no additional knowledge other than the word senses proper and morphological query expansions. We evaluate our method on two tasks. First, we use the acquired dominant senses to disambiguate the meanings of words in the Senseval-2 (Palmer et al., 2001) and Senseval-3 (Snyder and Palmer, 2004) data sets. Second, we simulate native speakers' intuitions about the salience of word meanings and examine whether the estimated sense frequencies correlate with sense production data. In all cases our approach outperforms a naive baseline and yields performances comparable to state of the art.

In the following section, we provide an overview of existing work on sense ranking. In Section 3, we introduce our IR-based method, and describe several sense ranking models. In Section 4, we present our results. Discussion of our results and future work conclude the paper (Section 5).

2 Related Work

McCarthy et al. (2004a) were the first to propose a computational model for acquiring dominant senses from text corpora. Key in their approach is the observation that distributionally similar neighbors often provide cues about a word's senses. The model quantifies the degree of similarity between a word's sense descriptions and its closest neighbors, thus delivering a ranking over senses where the most similar sense is intuitively the dominant sense. Their method exploits two notions of similarity, distributional and semantic. Distributionally similar words are acquired from the British National Corpus using an information-theoretic similarity measure (Lin, 1998) operating over dependency relations (e.g., verb-subject, verb-object). The latter are obtained from the output of Briscoe and Carroll's (2002) parser. The semantic similarity between neighbors and senses is measured using a manually crafted taxonomy such as WordNet (see Budanitsky and Hirst 2001 for an overview of WordNet-based similarity measures).

Mohammad and Hirst (2006) propose an algorithm for inferring dominant senses without relying on distributionally similar neighbors. Their approach capitalizes on the collocational nature of semantically related words. Assuming a coarse-grained sense inventory (e.g., the Macquarie Thesaurus), it first creates a matrix whose columns represent *all* categories (senses) $c_1 \dots c_n$ in the inventory and rows the ambiguous target words $w_1 \dots w_m$; the matrix cells record the number of times a target word t_i co-occurs with category c_j within a window of size s. Using an appropriate statistical test, they estimate the relative strength of association between an ambiguous word and each of its senses. The sense with the highest association is the predominant sense.

Our work shares with McCarthy et al. (2004a) and Mohammad and Hirst (2006) the objective of inferring dominant senses automatically. We propose a knowledge-lean method that relies on word association and requires no syntactic annotation. The latter may be unavailable when working with languages other than English for which state-of-the-art parsers or taggers have not been developed. Mohammad and Hirst (2006) estimate the co-occurrence frequency of a word and its sense descriptors by considering small window sizes of up to five words. These estimates will be less reliable for moderately frequent words or for sense inventories with many senses. Our approach is more robust to sparse data – we work with document-based frequencies – and thus suitable for both coarse and fine grained sense inventories. Furthermore, it is computationally inexpensive; in contrast to McCarthy et al. (2004a) we do not rely on the structure of the sense inventory for measuring the similarity between synonyms and their senses. Moreover, unlike Mohammad and Hirst (2006), our algorithm only requires co-occurrence frequencies for the target word and its senses, without considering all senses in the inventory and all words in the corpus simultaneously.

3 Method

3.1 Motivation

Central in our approach is the assumption that context provides important cues regarding a word's meaning. The idea dates back at least to Firth (1957) ("You shall know a word by the company it keeps") and underlies most WSD work to date. Another observation that has found wide application in WSD is that words tend to exhibit only one sense in a given discourse or document (Gale et al., 1992). Furthermore, documents are typically written with certain topics in mind which are often indicated by word distributional patterns (Harris, 1982).

For example, documents talking about congressional tenure are likely to contain words such as *term of office* or *incumbency*, whereas documents talking about legal tenure (i.e., the right to hold property) are likely to include the words *right* or *land*. Now, we could estimate which sense of tenure is most prevalent simply by comparing whether *tenure* co-occurs more often with *term of office* than with *land* provided we knew that both of these terms are semantically related to *tenure*. Fortunately, senses in WordNet (and related taxonomies) are represented by synonym terms. So, all we need to do for estimating a word's sense frequencies is to count how often it co-occurs with its synonyms. We adopt here a fairly broad definition of co-occurrence, two words co-occur if they are attested in the same document. We could obtain such counts from any document collection; however, to facilitate comparisons with prior work (e.g., McCarthy et al. 2004a), all our experiments use the British National Corpus (BNC). In what follows we describe in detail how we retrieve co-occurrence counts from the BNC and how we acquire dominant senses.

3.2 Dominant Sense Acquisition

Throughout the paper we use the term frequency as a shorthand for document frequency, i.e., the number of documents that contain a word or a set of words which may or may not be adjacent. The method we propose here exploits document frequencies of words and their sense definitions. We base our discussion below on the WordNet sense inventory and its representation of senses in terms of synonym sets (synsets). However, our approach is not limited to this particular lexicon; any dictionary with synonym-based sense definitions could serve our purposes.

As an example consider the noun *tenure*, which has the following senses in WordNet:

(1)
```
Sense 1
tenure, term of office, incumbency
=> term
Sense 2
tenure, land tenure
=> legal right
```

The senses are represented by the two synsets {`tenure, term of office, incumbency`} and {`tenure, land tenure`}. (The hypernyms for each sense are also listed; indicated by the arrows.) We can now approximate the frequency with which a word w_1 occurs with the sense s by computing its **synonym frequencies:** for each word $w_2 \in \text{syns}(s)$, the set of synonyms of s, we field a query of the form w_1 `AND` w_2. These synonym frequencies can then be used to determine the most frequent sense of w_1 in a variety of ways (to be detailed below).

The synsets for the two senses in (1) give rise to the queries in (2) and (3). Note that two queries are generated for the first synset, as it contains two synonyms of the target word *tenure*.

(2) a. `"tenure" AND "term of office"`
b. `"tenure" AND "incumbency"`

(3) `"tenure" AND "land tenure"`

For example, query (2-a) will return the number of documents in which *tenure* and *term of office* co-occur. Presumably, *tenure* is mainly used in its dominant sense in these documents. In the same way, query (3) will return documents in which *tenure* is used in the sense of *land tenure*. Note that this way of approximating synonym frequencies as document frequencies crucially relies on the "one sense per discourse" hypothesis (Gale et al., 1992), under the assumption that a document counts as a discourse for word sense disambiguation purposes.

Apart from synonym frequencies, we also generate **hypernym frequencies** by submitting queries of the form w_1 `AND` w_2, for each $w_2 \in \text{hype}(s)$, the set of immediate hypernyms of the sense s. The hypernym queries for the two senses of *tenure* are:

(4) `"tenure" AND "term"`

(5) `"tenure" AND "legal right"`

Hypernym queries are particularly useful for synsets of size one, i.e., where a word in a given sense has no synonyms, and is only differentiated from other senses by its hypernyms.

Before submitting queries such as the ones in (2) and (3) to an IR engine, we perform **query expansion** to make sure that all relevant inflected forms are included. For example the query term `"tenure"` is expanded to `("tenure" OR "tenures")`, i.e., both singular and plural noun forms are generated. Similarly, all inflected verb forms are generated, e.g., `"keep up"` gives rise to the query term `("keep up" OR "keeps up" OR "keeping up" OR "kept up")`. John Carroll's suite of morphological tools (`morpha` and `morphg`) is used to generate inflected forms for verbs and nouns.[1]

The queries generated this way are then submitted to an IR engine to obtain document counts. Specifically, we indexed the BNC using GLIMPSE (Global Implicit Search) a fast and flexible indexing and query system[2] (Manber and Wu, 1994). GLIMPSE supports approximate and exact matching, Boolean queries, wild cards, regular expressions, and many other options. The text is divided into equal size blocks and an inverted index is created containing the words and the block numbers in which they occur. Given a query, GLIMPSE will retrieve the relevant documents using a two-level search method. It will first locate the query in the inverted index and then use sequential search to find an exact answer.

Once synonym frequencies and hypernym frequencies are in place, we can compute a word's predominant sense in a number of ways. First, we can vary the way the frequency of a given sense is estimated based on synonym frequencies:

- **Sum:** The frequency of a given synset is computed as the sum of the synonym frequencies. For example, the frequency of the dominant sense of *tenure* would be computed by adding up the document frequencies returned by queries (2-a) and (2-b).

- **Average (Avg):** The frequency of a synset is computed by taking the average of synonym frequencies.

- **Highest (High):** The frequency of a synset is determined by the synonym with the highest frequency.

Secondly, we can vary whether or not hypernyms are taken into account:

- **No hypernyms ($-$Hyp):** Only the synonym frequencies are included when computing the frequency of a synset. For example, only the queries of (2-a) and (2-b) are relevant for estimating the dominant sense of *tenure*.

- **Hypernyms ($+$Hyp):** Both synonym and hypernym frequencies are taken into account when computing sense frequency. For example, the frequency for the senses of *tenure* would be computed based on the document frequencies returned by queries (2-a), (2-b), and (4) (by summing, averaging, or taking the highest value, as before).

The third option relates to whether the sense frequencies are used in raw or in normalized form:

- **Non-normalized ($-$Norm)**: The raw synonym frequencies are used as estimates of sense frequencies.

- **Normalized ($+$Norm)**: Sense frequencies are computed by dividing the word-synonym frequency by the frequency of the synonym in isolation. For example, the normalized frequency for (2-a) is computed by dividing the document frequency for `"tenure" AND "term of office"` by the document frequency for `"term of office"`. Normalizing takes into account the fact that the members of the synset of a sense may differ in frequency.

The combination of the above parameters yields 12 sense ranking models. We explore the parameter space exhaustively on the Senseval-2 benchmark data set. The best performing model on this data set is then used in all our subsequent experiments. We use Senseval-2 as a development set, but we also demonstrate that a far smaller manually annotated sample is sufficient for selecting the best model.

4 Experiments

Our experiments were driven by three questions: (1) Is WSD feasible at all with a model that does not employ any syntactic or semantic knowledge? Recall that McCarthy et al. (2004a) propose a model that crucially relies on a robust parser for estimating dominant senses. (2) What is the best parameter setting for our model? (3) Do the acquired dominant senses correlate with human judgments? If our sense frequencies exhibit no such correlation, it is unlikely that they will be useful in practical applications.

To address the first two questions we use the induced first senses to perform WSD on the Senseval-2 and Senseval-3 data sets. For our third question we compare native speakers' semantic intuitions against the BNC sense frequencies.

[1]The tools can be downloaded from `http://www.informatics.susx.ac.uk/research/nlp/carroll/morph.html`.

[2]The software can be downloaded from `http://webglimpse.net/download.php`

	−Norm				+Norm			
	+Hyp		−Hyp		+Hyp		−Hyp	
	P	*R*	*P*	*R*	*P*	*R*	*P*	*R*
Sum	42.3	40.8	46.3	44.6	45.9	44.3	48.6	46.8
High	51.6	49.8	51.1	49.3	57.2	55.1	**59.7**	**57.6**
Avg	44.1	42.6	48.5	46.8	49.6	47.8	51.5	49.6

Table 1: Results for Senseval-2 data by model instantiation

4.1 Model Selection

The goal of our first experiment is to establish which model configuration (see Section 3.2) is best suited for the WSD task. We thus varied how the overall frequency is computed (Sum, High, Avg), whether hyponyms are included (±Hyp), and whether the frequencies are normalized (±Norm). To explore the parameter space, we used the Senseval-2 all-words test data as our development set. This data set consists of three documents from the Wall Street Journal containing approximately 2,400 content words. Following McCarthy et al. (2004a), we first use our method to find the dominant sense for all word types in the corpus and then use that sense to disambiguate tokens without taking contextual information into account. We used WordNet 1.7.1 (Fellbaum, 1998) senses.[3]

We compared our results to a baseline that selects for each word type a random sense, assumes it is the dominant one, and uses it to disambiguate all instances of the target word (McCarthy et al., 2004a). We also report the WSD performance of a more competitive baseline that always chooses the sense with the largest synset as the dominant sense. Consider again the word *tenure* from Section 3.2. According to this baseline, the dominant sense for *tenure* is the first one since it is represented by the largest synset (three members).

Our results on Senseval-2 are summarized in Table 1. We observe that models that do not include hypernyms yield consistently better precision and recall than models that include them. On the one hand, hypernyms render the estimated sense distributions less sparse. On the other hand, they introduce considerable noise; the resulting sense frequencies are often similar – the same hypernyms can be shared among several senses – and selecting one predominant sense over the other can be due to very small frequency differences. We also find that models with normalized document counts outperform models without normalization. This is not surprising, there is ample evidence in the literature (Mohammad and Hirst, 2006; Turney, 2001) that association measures (e.g., conditional probability, mutual information) are better indicators of lexical similarity than raw frequency. Finally, selecting the synonym with the highest frequency (and defaulting to its sense) achieves better results in comparison to averaging or summing over all synsets.

In sum, the best performing model is High, +Norm, −Hyp, achieving a precision of 59.7% and a recall of 57.9%. The results for this model are broken down by part of speech in Table 2. Here, we also include a comparison with the random baseline (BaseR) and a baseline that selects the dominant sense by synset size (BaseS). We observe that the optimal model significantly outperforms both baselines on the complete data set (see row All in Table 2) and on most individual parts of speech (performances are comparable for our model and BaseS on adjectives). BaseS is far better than BaseR and generally harder to beat. Defaulting to synset size in the absence of any other information is a good heuristic; large synsets often describe frequent senses. Variants of our model that select a dominant sense by summing over synset members are closest to this baseline. Note that our best performing model does not rely on synset size; it simply selects the synonym with the highest frequency, despite the fact that it might belong to a large or small synset. We conjecture that its superior performance is due to the collocational nature of semantic similarity (Turney,

	BaseR		BaseS		Model		
	P	*R*	*P*	*R*	*P*	*R*	*N*
Noun	26.8	25.4	45.8	43.4	$53.1^{*\#}$	$50.2^{*\#}$	1,063
Verb	11.2	11.1	19.9	19.5	$48.2^{*\#}$	$47.3^{*\#}$	569
Adj	22.1	21.4	56.5	56.0	56.7^{*}	56.2^{*}	451
Adv	48.0	45.9	66.4	62.9	$86.4^{*\#}$	$81.8^{*\#}$	301
All	26.3	25.4	42.2	40.7	$59.7^{*\#}$	$57.6^{*\#}$	2,384

Table 2: Results of best model (High, +Norm, −Hyp) for Senseval-2 data by part of speech (*: sig. diff. from BaseR, #: sig. diff. from BaseS; $p < 0.01$ using χ^2 test)

[3]Senseval-2 is annotated with WordNet 1.7 senses which we converted to 1.7.1 using a publicly available mapping (see `http://www.cs.unt.edu/~rada/downloads.html`).

	−Norm				+Norm			
	+Hyp		−Hyp		+Hyp		−Hyp	
	P	*R*	*P*	*R*	*P*	*R*	*P*	*R*
Sum	42.3	40.8	46.3	44.6	45.2	44.7	44.6	44.0
High	51.6	49.8	51.1	49.3	55.0	54.3	**61.3**	**60.5**
Avg	44.1	42.6	48.5	46.8	51.5	50.8	50.4	49.8

Table 3: Results for 10% of Senseval-2 data by model instantiation

2001).

In order to establish that High, +Norm, −Hyp is the optimal model, we utilized the whole Senseval-2 data set. Using such a large dataset is more likely to yield a stable parameter setting, but it also raises the question whether parameter optimization could take place on a smaller dataset which is less costly to produce. Table 3 explores the parameter space on a sample randomly drawn from Senseval-2 that contains only 240 tokens (i.e., one tenth of the original data set). The behavior of our models on this smaller sample is comparable to that on the entire Senseval-2 data. Importantly, both sets yield the same best model, i.e., High, +Norm, −Hyp. In the remainder of this paper we will use this model for further experiments without additional parameter tuning.

4.2 Application to Senseval-3 Data

We next evaluate our best model the on the Senseval-3 English all-words data set. Senseval-3 consists of two Wall Street Journal articles and one excerpt from the Brown corpus (approximately 5,000 content words in total). Similarly to the experiments reported in the previous section, we used WordNet 1.7.1. We calculate recall and precision with the Senseval-3 scorer.

Our results are given in Table 4. Besides the two baselines (BaseR and BaseS), we also compare our model to McCarthy et al. (2004b)[4] and the best unsupervised (IRST-DDD) and supervised (GAMBLE) systems that participated in Senseval-3. IRST-DDD was developed by Strapparava et al. (2004) and performs domain driven disambiguation. Specifically, the approach compares the domain of the context surrounding the target word with the domains of its senses and uses a version of WordNet

	P	*R*
BaseR	$23.1^{\#\dagger\$\ddagger}$	$22.7^{\#\dagger\$\ddagger}$
BaseS	$36.6^{*\dagger\$\ddagger}$	$35.9^{*\dagger\$\ddagger}$
McCarthy	$49.0^{*\#\$\ddagger}$	$43.0^{*\#\$\ddagger}$
IR-Model	$58.0^{*\#\dagger\ddagger}$	$57.0^{*\#\dagger\ddagger}$
IRST-DDD	$58.3^{*\#\dagger\ddagger}$	$58.2^{*\#\dagger\ddagger}$
Semcor	$62.4^{*\#\dagger\$}$	$62.4^{*\#\dagger\$}$
GAMBLE	$65.1^{*\#\dagger\$\ddagger}$	$65.2^{*\#\dagger\$\ddagger}$

Table 4: Comparison of results on Senseval-3 data (*: sig. diff. from BaseR, $^\#$: sig. diff. from BaseS, †: sig. diff. from McCarthy, $^\$$: sig. diff. from IR-Model, ‡: sig. diff. from SemCor; $p < 0.01$ using χ^2 test)

	BaseR		BaseS		Model		
	P	*R*	*P*	*R*	*P*	*R*	*N*
Noun	27.8	12.2	41.1	41.0	$58.1^{*\#}$	$58.0^{*\#}$	900
Verb	12.8	4.6	20.0	19.9	$61.0^{*\#}$	$60.8^{*\#}$	732
Adj	29.2	5.2	56.5	56.5	50.3^{*}	50.3^{*}	363
Adv	100.0	0.6	100.0	81.2	100.0	81.2	16
All	23.1	22.7	36.6	35.9	$58.0^{*\#}$	$57.0^{*\#}$	2,011

Table 5: Results of best model (High, +Norm, −Hyp) for Senseval-3 data by part of speech (*: sig. diff. from BaseR, $^\#$: sig. diff. from BaseS; $p < 0.01$ using χ^2 test)

augmented with domain labels (e.g., economy, geography). GAMBL (Decadt et al., 2004) is a supervised system: a classifier is trained for each ambiguous word using memory-based learning. We also report the performance achieved by defaulting to the first WordNet entry for a given word and part of speech. Entries in WordNet are ranked according to the sense frequency estimates obtained from the manually annotated SemCor corpus. First senses obtained from SemCor will be naturally less noisy than those computed by our method which does not make use of manual annotation in any way. We therefore consider the WSD performance achieved with SemCor first senses as an upper bound for automatically acquired first senses.

Our model significantly outperforms the two baselines and McCarthy et al. (2004b). Its precision and recall according to individual parts of speech is shown in Table 5. The model performs comparably to IRST-DDD and significantly worse than GAMBLE. This is not entirely surprising given that GAM-

[4]Comparison against Mohammad and Hirst (2006) was not possible since they use a sense inventory other than WordNet (i.e., Roget's thesaurus) and evaluate their model on artificially generated sense-tagged data.

BLE is a supervised system trained on a variety of manually annotated resources including SemCor, data from previous Senseval workshops and the example sentences in WordNet 1.7.1. GAMBLE is the only system that significantly outperforms the SemCor upper bound. Finally, note that our model is conceptually simpler than McCarthy et al. (2004b) and IRST-DDD. It neither requires a parser (for obtaining distributionally similar neighbors) nor any knowledge other than WordNet (e.g., domain labels). This makes our method portable to languages for which syntactic analysis tools and elaborate semantic resources are not available.

4.3 Modeling Human Data

Research in psycholinguistics has shown that the meanings of ambiguous words are not perceived as equally salient in the absence of a biasing context (Durkin and Manning, 1989; Twilley et al., 1994). Rather, language users often ascribe dominant and subordinate meanings to polysemous words. Previous studies have elicited intuitions with regard to word senses using a free association task. For example, Durkin and Manning (1989) collected association norms from native speakers for 175 ambiguous words. They asked subjects to read each word and write down the first meaning that came to mind. The words were presented out of context. From the subjects' responses, they computed sense frequencies, which revealed that most words were attributed a particular meaning with a markedly higher frequency than other meanings.

In this experiment, we examine whether our model agrees with human intuitions regarding the prevalence of word senses. We inferred the dominant meanings for the polysemous words used in Durkin and Manning (1989). These exhibit a relatively high degree of ambiguity (the average number of senses per word is three) and cover a wide variety of parts of speech (for the full set of words and elicited sense frequencies see their Appendix A, pp. 501–609). One stumbling block to using this data are the meanings associated with the ambiguous words. These were provided by native English speakers and may not necessarily correspond to senses described by trained lexicographers. Fortunately, we were able to map most of them (except for six which we discarded) on WordNet synsets (version 1.6); two annotators performed the mapping by comparing the sense descriptions provided by Durkin and Manning to WordNet synsets. The annotators agreed in their assignments 81% of the time. Disagreements were resolved through mediation.

act	Freq
pretense/performance	37
to perform	30
to take action	16
division	12
a deed	3

answer	Freq
response	81
solution	18

Table 6: Meaning frequencies for *act* and *answer*; normative data from Durkin and Manning (1989)

Examples of Durkin and Manning's (1989) normative data are given in Table 6. The sense `response` for *answer* was mapped to the WordNet synset {`answer, reply, response`} (Sense 1), the sense `solution` was mapped to the synset {`solution, answer, result, resolution, solvent`} (Sense 2), etc. Durkin and Manning did not take part of speech ambiguity into account, as Table 6 shows, subjects came up with meanings relating to the verb and noun part of speech of *act*.

We explored the relationship between the sense frequencies provided by human subjects and those estimated by our model by computing the Spearman rank correlation coefficient ρ. We obtained sense frequencies from the BNC using the best model from Section 4.1 (High, +Norm, −Hyp). We found that the resulting sense frequencies were significantly correlated with the human sense frequencies ($\rho = 0.384$, $p < 0.01$). We performed the same experiment using McCarthy et al.'s (2004a) model, which also achieved a significant correlation ($\rho = 0.316$, $p < 0.01$). This result provides an additional validation of our model as it demonstrates that the sense frequencies it generates can capture the sense preferences of naive human subjects (rather than trained lexicographers).

5 Discussion

In this paper we proposed an IR-based approach for inducing dominant senses automatically. Our method estimates the degree of association between words and their sense descriptions (represented by synsets in WordNet) simply by querying an IR engine. Evaluation on the Senseval data sets showed that our model significantly outperformed a naive random sense baseline and a more competitive one based on synset size. Our method was significantly better than McCarthy et al. (2004b) on Senseval-2 and Senseval-3. On the latter data set, its performance was comparable to that of the best unsupervised system (Strapparava et al., 2004).

An important future direction lies in evaluating the disambiguation potential of our models across domains and languages. Furthermore, our experiments have relied on WordNet for providing the appropriate sense descriptions. Future work must assess whether the models presented in this paper can be extended to alternative sense inventories (e.g., dictionary definitions) that may differ in granularity and structure. We will also experiment with a wider range of lexical association measures for quantifying the similarity of a word and its synonyms. Examples include odds ratio (Mohammad and Hirst, 2006) and Turney's (2001) IR-based pointwise mutual information (PMI-IR).

Our experiments revealed that the IR-based model is particularly good at disambiguating certain parts of speech (e.g., verbs, see Tables 2 and 5). A promising direction is the combination of different ranking models (Brody et al., 2006) and the integration of dominant sense models with supervised WSD.

Acknowledgments We are grateful to Diana McCarthy for her help with this work. The authors acknowledge the support of EPSRC (grant EP/C538447/1).

References

Briscoe, Ted and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*. Las Palmas, Gran Canaria, pages 1499–1504.

Brody, Samuel, Roberto Navigli, and Mirella Lapata. 2006. Ensemble methods for unsupervised WSD. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 97–104.

Budanitsky, Alexander and Graeme 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, PA.

Decadt, Bart, Véronique Hoste, Walter Daelemans, and Antal van den Bosch. 2004. GAMBL, genetic algorithm optimization of memory-based WSD. In Mihalcea and Edmonds (2004), pages 108–112.

Durkin, Kevin and Jocelyn Manning. 1989. Polysemy and the subjective lexicon: Semantic relatedness and the salience of intraword senses. *Journal of Psycholinguistic Research* 18(6):577–612.

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

Firth, J. R. 1957. *A Synopsis of Linguistic Theory 1930-1955*. Oxford: Philological Society.

Gale, William A., Kenneth W. Church, and David Yarowsky. 1992. A method for disambiguating word senses in a large corpus. *Computers and the Humanities* 26(5–6):415–439.

Harris, Zellig. 1982. Discourse and sublanguage. In R. Kittredge and J. Lehrberger, editors, *Language in Restricted Semantic Domains*, Walter de Gruyter, Berlin; New York, pages 231–236.

Lin, Dekang. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*. Madison, WI, pages 296–304.

Manber, Udi and Sun Wu. 1994. GLIMPSE: a tool to search through entire file systems. In *Proceedings of USENIX Winter 1994 Technical Conference*. San Francisco, CA, pages 23–32.

McCarthy, Diana, Rob Koeling, Julie Weeds, and John Carroll. 2004a. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*. Barcelona, pages 279–286.

McCarthy, Diana, Rob Koeling, Julie Weeds, and John Carroll. 2004b. Using automatically acquired predominant senses for word sense disambiguation. In Mihalcea and Edmonds (2004), pages 151–154.

Mihalcea, Rada and Phil Edmonds, editors. 2004. *Proceedings of Senseval-3: The 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Barcelona.

Mohammad, Saif and Graeme Hirst. 2006. Determining word sense dominance using a thesaurus. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy, pages 121–128.

Palmer, Martha, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All words and verb lexical sample. In *Proceedings of Senseval-2: The 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Toulouse.

Snyder, Benjamin and Martha Palmer. 2004. The English all-words task. In Mihalcea and Edmonds (2004).

Stokoe, Christopher. 2005. Differentiating homonymy and polysemy in information retrieval. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*. Vancouver, pages 403–410.

Strapparava, Carlo, Alfio Gliozzo, and Claudio Giuliano. 2004. Word-sense disambiguation for machine translation. In Mihalcea and Edmonds (2004), pages 229–234.

Turney, Peter D. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*. Freiburg, Germany, pages 491–502.

Twilley, L. C., P. Dixon, D. Taylor, and K. Clark. 1994. University of Alberta norms of relative meaning frequency for 566 homographs. *Memory and Cognition* 22(1):111–126.

Vickrey, David, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*. Vancouver, pages 771–778.

Yarowsky, David and Radu Florian. 2002. Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering* 9(4):293–310.

Near-Synonym Choice in an Intelligent Thesaurus

Diana Inkpen
School of Information Technology and Engineering,
University of Ottawa
800 King Edward, Ottawa, ON, Canada, K1N 6N5
diana@site.uottawa.ca

Abstract

An intelligent thesaurus assists a writer with alternative choices of words and orders them by their suitability in the writing context. In this paper we focus on methods for automatically choosing near-synonyms by their semantic coherence with the context. Our statistical method uses the Web as a corpus to compute mutual information scores. Evaluation experiments show that this method performs better than a previous method on the same task. We also propose and evaluate two more methods, one that uses anti-collocations, and one that uses supervised learning. To asses the difficulty of the task, we present results obtained by human judges.

1 Introduction

When composing a text, a writer can access a thesaurus to retrieve words that are similar to a given target word, when there is a need to avoid repeating the same word, or when the word does not seem to be the best choice in the context.

Our intelligent thesaurus is an interactive application that presents the user with a list of alternative words (near-synonyms), and, unlike standard thesauri, it orders the choices by their suitability to the writing context. We investigate how the collocational properties of near-synonyms can help with choosing the best words. This problem is difficult because the near-synonyms have senses that are very close to each other, and therefore they occur in similar contexts; we need to capture the subtle differences specific to each near-synonym.

Our thesaurus brings up only alternatives that have the same part-of-speech with the target word. The choices could come from various inventories of near-synonyms or similar words, for example the Roget thesaurus (Roget, 1852), dictionaries of synonyms (Hayakawa, 1994), or clusters acquired from corpora (Lin, 1998).

In this paper we focus on the task of automatically selecting the best near-synonym that should be used in a particular context. The natural way to validate an algorithm for this task would be to ask human readers to evaluate the quality of the algorithm's output, but this kind of evaluation would be very laborious. Instead, we validate the algorithms by deleting selected words from sample sentences, to see whether the algorithms can restore the missing words. That is, we create a *lexical gap* and evaluate the ability of the algorithms to *fill the gap*. Two examples are presented in Figure 1. All the near-synonyms of the original word, including the word itself, become the choices in the solution set (see the figure for two examples of solution sets). The task is to automatically fill the gap with the best choice in the particular context. We present a method of scoring the choices. The highest scoring near-synonym will be chosen. In order to evaluate how well our method works we consider that the only correct solution is the original word. This will cause our evaluation scores to underestimate the performance, as more than one choice will sometimes be a perfect

Sentence: This could be improved by more detailed consideration of the processes of .... propagation inherent in digitizing procedures.
Original near-synonym: error
Solution set: mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism

Sentence: The day after this raid was the official start of operation strangle, an attempt to completely destroy the .... lines of communication.
Original near-synonym: enemy
Solution set: opponent, adversary, antagonist, competitor, enemy, foe, rival

Figure 1: Examples of sentences with a lexical gap, and candidate near-synonyms to fill the gap.

solution. Moreover, what we consider to be the best choice is the typical usage in the corpus, but it may vary from writer to writer. Nonetheless, it is a convenient way of producing test data.

The statistical method that we propose here is based on semantic coherence scores (based on mutual information) of each candidate with the words in the context. We explore how far such a method can go when using the Web as a corpus. We estimate the counts by using the Waterloo MultiText[1] system (Clarke and Terra, 2003b) with a corpus of about one terabyte of text collected by a Web crawler. We also propose a method that uses collocations and anti-collocations, and a supervised method that uses words and mutual information scores as featured for machine learning.

2 Related work

The idea of using the Web as a corpus of texts has been exploited by many researchers. Grefenstette (1999) used the Web for example-based machine translation; Kilgarriff (2001) investigated the type of noise in Web data; Mihalcea and Moldovan (1999) and Agirre and Martinez (2000) used it as an additional resource for word sense disambiguation; Resnik (1999) mined the Web for bilingual texts; Turney (2001) used Web frequency counts to compute information retrieval-based mutual-information scores. In a *Computational Linguistics* special issue on the Web as a corpus (Kilgarriff and Grefenstette, 2003), Keller and Lapata (2003) show that Web counts correlate well with counts collected from a balanced corpus: the size of the Web compensates for the noise in the data. In this paper we are using a very large corpus of Web pages to address a problem that has not been successfully solved before.

In fact, the only work that addresses exactly the same task is that of Edmonds (1997), as far as we are aware. Edmonds gives a solution based on a lexical co-occurrence network that included second-order co-occurrences. We use a much larger corpus and a simpler method, and we obtain much better results.

Our task has similarities to the word sense disambiguation task. Our near-synonyms have senses that are very close to each other. In Senseval, some of the fine-grained senses are also close to each other, so they might occur in similar contexts, while the coarse-grained senses are expected to occur in distinct contexts. In our case, the near-synonyms are different words to choose from, not the same word with different senses.

3 A statistical method for near-synonym choice

Our method computes a score for each candidate near-synonym that could fill in the gap. The near-synonym with the highest score is the proposed solution. The score for each candidate reflects how well a near-synonym fits in with the context. It is based on the mutual information scores between a near-synonym and the content words in the context (we filter out the stopwords).

The **pointwise mutual information** (PMI) between two words x and y compares the probability of observing the two words together (their joint probability) to the probabilities of observing x and y independently (the probability of occurring together by chance) (Church and Hanks, 1991): $\mathrm{PMI}(x, y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$

The probabilities can be approximated by: $P(x) = C(x)/N$, $P(y) = C(y)/N$, $P(x, y) = C(x, y)/N$, where C denotes frequency counts and N is the total number of words in the corpus. Therefore: $\mathrm{PMI}(x, y) = \log_2 \frac{C(x,y)\cdot N}{C(x)\cdot C(y)}$, where N can be ignored in comparisons.

We model the context as a window of size $2k$

[1]We thank Egidio Terra, Charlie Clarke, and Univ. of Waterloo for allowing us to use MultiText, and to Peter Turney and IIT/NRC for giving us access to their local copy of the corpus.

around the gap (the missing word): k words to the left and k words to the right of the gap. If the sentence is $s = \cdots w_1 \cdots w_k \;\; Gap \;\; w_{k+1} \cdots w_{2k} \cdots$, for each near-synonym NS_i from the group of candidates, the semantic coherence score is computed by the following formula:

$$Score(NS_i, s) = \Sigma_{j=1}^{k} \text{PMI}(NS_i, w_j) + \Sigma_{j=k+1}^{2k} \text{PMI}(NS_i, w_j).$$

We also experimented with the same formula when the sum is replaced with maximum to see whether a particular word in the context has higher influence than the sum of all contributions (though the sum worked better).

Because we are using the Waterloo terabyte corpus and we issue queries to its search engine, we have several possibilities of computing the frequency counts. $C(x, y)$ can be the number of co-occurrences of x and y when y immediately follows x, or the distance between x and y can be up to q. We call q the query frame size. The tool for accessing the corpus allows us to use various values for q.

The search engine also allows us to approximate words counts with document counts. If the counts $C(x)$, $C(y)$, and $C(x, y)$ are approximated as the number of document in which they appear, we obtain the PMI-IR formula (Turney, 2001). The queries we need to send to the search engine are the same but they are restricted to document counts: $C(x)$ is the number of document in which x occurs; $C(x, y)$ is the number of documents in which x is followed by y in a frame of size q.

Other statistical association measures, such as log-likelihood, could be used. We tried only PMI because it is easy to compute on a Web corpus and because PMI performed better than other measures in (Clarke and Terra, 2003a).

We present the results in Section 6.1, where we compare our method to a baseline algorithm that always chooses the most frequent near-synonyms and to Edmonds's method for the same task, on the same data set. First, however, we present two other methods to which we compare our results.

4 The anti-collocations method

For the task of near-synonym choice, another method that we implemented is the anti-collocations method. By *anti-collocation* we mean a combination of words that a native speaker would not use and therefore should not be used when automatically generating text. This method uses a knowledge-base of collocational behavior of near-synonyms acquired in previous work (Inkpen and Hirst, 2006). A fragment of the knowledge-base is presented in Table 1, for the near-synonyms of the word *error* and two collocate words *ghastly* and *spelling*. The lines marked by $*$ represent anti-collocations and the rest represent strong collocations.

ghastly mistake	spelling mistake
$*$ghastly error	spelling error
ghastly blunder	$*$spelling blunder
$*$ghastly faux pas	$*$spelling faux pas
$*$ghastly blooper	$*$spelling blooper
$*$ghastly solecism	$*$spelling solecism
$*$ghastly goof	$*$spelling goof
$*$ghastly contretemps	$*$spelling contretemps
$*$ghastly boner	$*$spelling boner
$*$ghastly slip	$*$spelling slip

Table 1: Examples of collocations and anti-collocations. The $*$ indicates the anti-collocations.

The anti-collocations method simply ranks the strong collocations higher than the anti-collocations. In case of ties it chooses the most frequent near-synonym. In Section 6.2 we present the results of comparing this method to the method from the previous section.

5 A supervised learning method

We can also apply supervised learning techniques to our task. It is easy to obtain labeled training data, the same way we collected test data for the two unsupervised methods presented above. We train classifiers for each group of near-synonyms. The classes are the near-synonyms in the solution set. The word that produced the gap is the expected solution, the class label; this is a convenient way of producing training data, no need for manual annotation. Each sentence is converted into a vector of features to be used for training the supervised classifiers. We used two types of features. The features of the first type are the PMI scores of the left and right context with each class (each near-synonym from the group). The number of features of this type is twice the number of classes, one score for the part of the sentence at the left of the gap, and one for the part at the right of the gap. The features of the second type are the

1. mistake, error, fault
2. job, task, chore
3. duty, responsibility, obligation
4. difficult, hard
5. material, stuff
6. put up, provide, offer
7. decide, settle, resolve, adjudicate.

Table 2: The near-synonym groups used in Exp1.

words in the context window. For each group of near-synonyms, we used as features the 500 most-frequent words situated close to the gaps in a development set. The value of a word feature for each training example is 1 if the word is present in the sentence (at the left or at the right of the gap), and 0 otherwise. We trained classifiers using several machine learning algorithms, to see which one is best at discriminating among the near-synonyms. In Section 6.3, we present the results of several classifiers.

A disadvantage of the supervised method is that it requires training for each group of near-synonyms. Additional training would be required whenever we add more near-synonyms to our knowledge-base.

6 Evaluation

6.1 Comparison to Edmonds's method

In this section we present results of the statistical method explained in Section 3. We compare our results with those of Edmonds's (1997), whose solution used the texts from the year 1989 of the Wall Street Journal (WSJ) to build a lexical co-occurrence network for each of the seven groups of near-synonyms from Table 2. The network included second-order co-occurrences. Edmonds used the WSJ 1987 texts for testing, and reported accuracies only a little higher than the baseline. The near-synonyms in the seven groups were chosen to have low polysemy. This means that some sentences with wrong senses of near-synonyms might be in the

For comparison purposes, in this section we use the same test data (WSJ 1987) and the same groups of near-synonyms (we call these sentences the Exp1 data set). Our method is based on mutual information, not on co-occurrence counts. Our counts are collected from a much larger corpus.

Table 3 presents the comparative results for the seven groups of near-synonyms (we did not repeat

		Accuracy			
Set	No. of cases	Base-line	Edmonds method	Stat. method (Docs)	Stat. method (Words)
1.	6,630	41.7%	47.9%	**61.0**%	59.1%
2.	1,052	30.9%	48.9%	**66.4**%	61.5%
3.	5,506	70.2%	68.9%	69.7%	**73.3**%
4.	3,115	38.0%	45.3%	64.1%	**66.0**%
5.	1,715	59.5%	64.6%	68.6%	**72.2**%
6.	11,504	36.7%	48.6%	52.0%	**52.7**%
7.	1,594	37.0%	65.9%	74.5%	**76.9**%
AVG	31,116	44.8%	55.7%	65.1%	**66.0**%

Table 3: Comparison between the statistical method from Section 3, baseline algorithm, and Edmonds's method (Exp1 data set).

them in the first column of the table, only the number of the group.). The last row averages the accuracies for all the test sentences. The second column shows how many test sentences we collected for each near-synonym group. The third column is for the baseline algorithm that always chooses the most frequent near-synonym. The fourth column presents the results reported in (Edmonds, 1997). column show the results of the supervised learning classifier described in Section 5. The fifth column presents the result of our method when using document counts in PMI-IR, and the last column is for the same method when using word counts in PMI. We show in bold the best accuracy for each data set. We notice that the automatic choice is more difficult for some near-synonym groups than for the others. In this paper, by accuracy we mean the number of correct choices made by each method (the number of gaps that were correctly filled). The correct choice is the near-synonym that was initially replaced by the gap in the test sentence.

To fine-tune our statistical method, we used the data set for the near-synonyms of the word *difficult* collected from the WSJ 1989 corpus as a development set. We varied the context window size k and the query frame q, and determined good values for the parameter k and q. The best results were obtained for small window sizes, $k = 1$ and $k = 2$ (meaning k words to the left and k words to the right of the gap). For each k, we varied the query frame size q. The results are best for a relatively small query frame, $q = 3, 4, 5$, when the query frame is the same or slightly larger then the context window.

The results are worse for a very small query frame, $q = 1, 2$ and for larger query frames $q = 6, 7, ..., 20$ or unlimited. The results presented in the rest of the paper are for $k = 2$ and $q = 5$. For all the other data sets used in this paper (from WSJ 1987 and BNC) we use the parameter values as determined on the development set.

Table 3 shows that the performance is generally better for word counts than for document counts. Therefore, we prefer the method that uses word counts (which is also faster in our particular setting). The difference between them is not statistically significant. Our statistical method performs significantly better than both Edmond's method and the baseline algorithm. For all the results presented in this paper, statistical significance tests were done using the paired t-test, as described in (Manning and Schütze, 1999), page 209.

On average, our method performs 22 percentage points better than the baseline algorithm, and 10 percentage points better than Edmonds's method. Its performance is similar to that of the supervised method (see Section 6.3). An important advantage of our method is that it works on any group of near-synonyms without training, whereas Edmonds's method required a lexical co-occurrence network to be built in advance for each group of near-synonyms and the supervised method required training for each near-synonym group.

We note that the occasional presence of near-synonyms with other senses than the ones we need might make the task somewhat easier. Nonetheless, the task is still difficult, even for human judges, as we will see in Section 6.4. On the other hand, because the solution allows only one correct answer the accuracies are underestimated.

6.2 Comparison to the anti-collocations method

In a second experiment we compare the results of our methods with the anti-collocation method described in Section 4. We use the data set from our previous work, which contain sentences from the first half of the British National Corpus, with near-synonyms from the eleven groups from Table 4.

The number of near-synonyms in each group is higher compared with WordNet synonyms, because they are taken from (Hayakawa, 1994), a dictionary

1. benefit, advantage, favor, gain, profit
2. low, gush, pour, run, spout, spurt, squirt, stream
3. deficient, inadequate, poor, unsatisfactory
4. afraid, aghast, alarmed, anxious, apprehensive, fearful, frightened, scared, terror-stricken
5. disapproval, animadversion, aspersion, blame, criticism, reprehension
6. mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism
7. alcoholic, boozer, drunk, drunkard, lush, sot
8. leave, abandon, desert, forsake
9. opponent, adversary, antagonist, competitor, enemy, foe, rival
10. thin, lean, scrawny, skinny, slender, slim, spare, svelte, willowy, wiry
11. lie, falsehood, fib, prevarication, rationalization, untruth

Table 4: The near-synonym groups used in Exp2.

that explains differences between near-synonyms. Moreover we retain only the sentences in which at least one of the context words is in our previously acquired knowledge-base of near-synonym collocations. That is, the anti-collocations method works only if we know how a word in the context collocates with the near-synonyms from a group. For the sentences that do not contain collocations or anti-collocations, it will perform no better than the baseline, because the information needed by the method is not available in the knowledge-base. Even if we increase the coverage of the knowledge-base, the anti-collocation method might still fail too often due to words that were not included.

Table 5 presents the results of the comparison. We used two data sets: TestSample, which includes at most two sentences per collocation (the first two sentences from the corpus); and TestAll, which includes all the sentences with collocations as they occurred in the corpus. The reason we chose these two tests is not to bias the results due to frequent collocations.

The last two columns are the accuracies achieved by our method. The second last column shows the results of the method when the word counts are approximated with document counts. The improvement over the baseline is 16 to 27 percentage points. The improvement over the anti-collocations method is 10 to 17 percentage points.

6.3 Comparison to supervised learning

We present the results of the supervised method from Section 5 on the data sets used in Section 6.1.

		Accuracy			
Test set	No. of cases	Base-line	Anti-collocs method	Stat. method (Docs)	Stat. method (Words)
Test Sample	171	57.0%	63.3%	**75.6%**	73.3%
TestAll	332	48.5%	58.6%	70.0%	**75.6%**

Table 5: Comparison between the statistical method from Section 3 and the anti-collocations method from Section 4. (Exp2 data set from Section 6.2).

ML method (Weka)	Features	Accuracy
Decision Trees	PMI scores	65.4%
Decision Rules	PMI scores	65.5%
Naïve Bayes	PMI scores	52.5%
K-Nearest Neighbor	PMI scores	64.5%
Kernel Density	PMI scores	60.5%
Boosting (Dec. Stumps)	PMI scores	**67.7%**
Naïve Bayes	500 words	**68.0%**
Decision Trees	500 words	67.0%
Naïve Bayes	PMI + 500 words	66.5%
Boosting (Dec. Stumps)	PMI + 500 words	**69.2%**

Table 6: Comparative results for the supervised learning method using various ML learning algorithms (Weka), averaged over the seven groups of near-synonyms from the Exp1 data set.

As explained before, the data sets contain sentences with a lexical gap. For each of the seven groups of near-synonyms, the class to choose from, in order to fill in the gaps is one of the near-synonyms in each cluster. We implemented classifiers that use as features either the PMI scores of the left and right context with each class, or the words in the context windows, or both types of features combined. We used as features the 500 most-frequent words for each group of near-synonyms. We report accuracies for 10-fold cross-validation.

Table 6 presents the results, averaged for the seven groups of near-synonyms, of several classifiers from the Weka package (Witten and Frank, 2000). The classifiers that use PMI features are Decision Trees, Decision Rules, Naïve Bayes, K-Nearest Neighbor, Kernel Density, and Boosting a weak classifier (Decision Stumps – which are shallow decision trees). Then, a Naïve Bayes classifier that uses only the word features is presented, and the same type of classifiers with both types of features. The other classifiers from the Weka package were also tried, but the results did not improve and these algorithms

	Accuracy			
Test set	Base-line	Supervised Boosting (PMI)	Supervised Boosting (PMI+words)	Unsuper-vised method
1.	41.7%	55.8%	57.3%	**59.1**%
2.	30.9%	68.1%	**70.8**%	61.5%
3.	70.2%	86.5%	**86.7**%	73.3%
4.	38.0%	66.5%	**66.7**%	66.0%
5.	59.5%	70.4%	71.0%	**72.2**%
6.	36.7%	53.0%	**56.1**%	52.7%
7.	37.0%	74.0%	75.8%	**76.9**%
AVG	44.8%	67.7%	**69.2**%	66.0%

Table 7: Comparison between the unsupervised statistical method from Section 3 and the supervised method described in Section 5, on the Exp1 data set.

had difficulties in scaling up. In particular, when using the 500 word features for each training example, only the Naïve Bayes algorithm was able to run in reasonable time. We noticed that the Naïve Bayes classifier performs very poorly on PMI features only (55% average accuracy), but performs very well on word features (68% average accuracy). In contrast, the Decision Tree classifier performs well on PMI features, especially when using boosting with Decision Stumps. When using both the PMI scores and the word features, the results are slightly higher. It seems that both types of features are sufficient for training a good classifier, but combining them adds value.

Table 7 presents the detailed results of two of the supervised classifiers, and repeats, for easier comparison, the results of the unsupervised statistical method from Section 6.1. The supervised classifier that uses only PMI scores performs similar to the unsupervised method. The best supervised classifier, that uses both types of features, performs slightly better than the unsupervised statistical method, but the difference is not statistically significant. We conclude that the results of the supervised methods and the unsupervised statistical method are similar. An important advantage of the unsupervised method is that it works on any group of near-synonyms without training.

6.4 Results obtained by human judges

We asked two human judges, native speakers of English, to guess the missing word in a random sample of the Exp1 data set (50 sentences for each of the

Test set	J1-J2 Agreement	J1 Acc.	J2 Acc.	System Accuracy
1.	72%	70%	76%	53%
2.	82%	84%	84%	68%
3.	86%	92%	92%	78%
4.	76%	82%	76%	66%
5.	76%	82%	74%	64%
6.	78%	68%	70%	52%
7.	80%	80%	90%	77%
AVG	78.5%	79.7%	80.2%	65.4%

Table 8: Results obtained by two human judges on a random subset of the Exp1 data set.

7 groups of near-synonyms, 350 sentences in total). The judges were instructed to choose words from the list of near-synonyms. The choice of a word not in the list was allowed, but not used by the two judges. The results in Table 8 show that the agreement between the two judges is high (78.5%), but not perfect. This means the task is difficult, even if some wrong senses in the test data might have made the task easier in a few cases.

The human judges were allowed to choose more than one correct answer when they were convinced that more than one near-synonym fits well in the context. They used this option sparingly, only in 5% of the 350 sentences. Taking the accuracy achieved of the human judges as an upper limit, the automatic method has room for improvement (10-15 percentage points). In future work, we plan to allow the system to make more than one choice when appropriate (for example when the second choice has a very close score to the first choice).

7 The intelligent thesaurus

Our experiments show that the accuracy of the first choice being the best choice is 66 to 75%; therefore there will be cases when the writer will not choose the first alternative. But the accuracy for the first two choices is quite high, around 90%, as presented in Table 9.

If the writer is in the process of writing and selects a word to be replaced with a near-synonym proposed by the thesaurus, then only the context on the left of the word can be used for ordering the alternatives. Our method can be easily adapted to consider only the context on the left of the gap. The results of this case are presented in Table 10, for the data sets

Test set	Accuracy first choice	Accuracy first 2 choices
Exp1, AVG	66.0%	**88.5%**
Exp2, TestSample	73.3%	**94.1%**
Exp2, TestAll	75.6%	**87.5%**

Table 9: Accuracies for the first two choices as ordered by an interactive intelligent thesaurus.

Test set	Accuracy first choice	Accuracy first 2 choices
Exp1, AVG	58.0%	**84.8%**
Exp2, TestSample	57.4%	**75.1%**
Exp2, TestAll	56.1%	**77.4%**

Table 10: Results of the statistical method when only the left context is considered.

used in the previous sections. The accuracy values are lower than in the case when both the left and the right context are considered (Table 9). This is due in part to the fact that some sentences in the test sets have very little left context, or no left context at all. On the other hand, many times the writer composes a sentence or paragraph and then she/he goes back to change a word that does not sound right. In this case, both the left and right context will be available.

In the intelligent thesaurus, we could combine the supervised and unsupervised method, by using a supervised classifier when the confidence in the classification is high, and by using the unsupervised method otherwise. Also the unsupervised statistical method would be used for the groups of near-synonyms for which a supervised classifier was not previously trained.

8 Conclusion

We presented a statistical method of choosing the best near-synonym in a context. We compared this method to a previous method (Edmonds's method) and to the anti-collocation method and showed that the performance improved considerably. We also show that the unsupervised statistical method performs comparably to a supervised learning method.

Our method based on PMI scores performs well, despite the well-known limitations of PMI in corpora. PMI tends to have problems mostly on very small counts, but it works reasonably with larger counts. Our web corpus is quite large, therefore the problem of small counts does not appear.

In the intelligent thesaurus, we do not make the near-synonym choice automatically, but we let the user choose. The first choice offered by the thesaurus is the best one quite often; the first two choices are correct 90% of the time.

Future work includes a word sense disambiguation module. In case the target word selected by the writer has multiple senses, they could trigger several groups of near-synonyms. The system will decide which group represents the most likely senses by computing the semantic coherence scores averaged over the near-synonyms from each group.

We plan to explore the question of which inventory of near-synonyms or similar words is the most suitable for use in the intelligent thesaurus.

Choosing the right near-synonym in context is also useful in other applications, such as natural language generation (NLG) and machine translation. In fact we already used the near-synonym choice module in an NLG system, for complementing the choices made by using the symbolic knowledge incorporated into the system.

References

Eneko Agirre and David Martinez. 2000. Exploring automatic word sense disambiguation with decision lists and the Web. In *Proceedings of the Workshop on Semantic Annotation And Intelligent Content, COLING 2000*, Saarbrücken/Luxembourg/Nancy.

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

Charles L. A. Clarke and Egidio Terra. 2003a. Frequency estimates for statistical word similarity measures. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 165–172, Edmonton, Canada.

Charles L. A. Clarke and Egidio Terra. 2003b. Passage retrieval vs. document retrieval for factoid question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 427–428, Toronto, Canada.

Philip Edmonds. 1997. Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 507–509, Madrid, Spain.

Gregory Grefenstette. 1999. The World Wide Web as a resource for example-based machine translation tasks. In *Proceedings of the ASLIB Conference on Translating and Computers*, London, UK.

S. I. Hayakawa, editor. 1994. *Choose the Right Word*. Second Edition, revised by Eugene Ehrlich. HarperCollins Publishers.

Diana Inkpen and Graeme Hirst. 2006. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32 (2):223–262.

Frank Keller and Mirella Lapata. 2003. Using the Web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29 (3):459–484.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the Web as a corpus. *Computational Linguistics*, 29 (3):333–347.

Adam Kilgarriff. 2001. Web as corpus. In *Proceedings of the 2001 Corpus Linguistics conference*, pages 342–345, Lancaster, UK.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics joint with 17th International Conference on Computational Linguistics (ACL-COLING'98)*, pages 768–774, Montreal, Quebec, Canada.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Rada Mihalcea and Dan Moldovan. 1999. A method for word sense disambiguation from unrestricted text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 152–158, Maryland, MD.

Philip Resnik. 1999. Mining the Web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 527–534, Maryland, MD.

Peter Mark Roget, editor. 1852. *Roget's Thesaurus of English Words and Phrases*. Longman Group Ltd., Harlow, Essex, UK.

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

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, CA.

A Log-linear Block Transliteration Model based on Bi-Stream HMMs

Bing Zhao, Nguyen Bach, Ian Lane, and **Stephan Vogel**
{bzhao, nbach, ianlane, vogel}@cs.cmu.edu
Language Technologies Institute
School of Computer Science, Carnegie Mellon University

Abstract

We propose a novel HMM-based framework to accurately transliterate unseen named entities. The framework leverages features in letter-alignment and letter n-gram pairs learned from available bilingual dictionaries. Letter-classes, such as vowels/non-vowels, are integrated to further improve transliteration accuracy. The proposed transliteration system is applied to out-of-vocabulary named-entities in statistical machine translation (SMT), and a significant improvement over traditional transliteration approach is obtained. Furthermore, by incorporating an automatic spell-checker based on statistics collected from web search engines, transliteration accuracy is further improved. The proposed system is implemented within our SMT system and applied to a real translation scenario from Arabic to English.

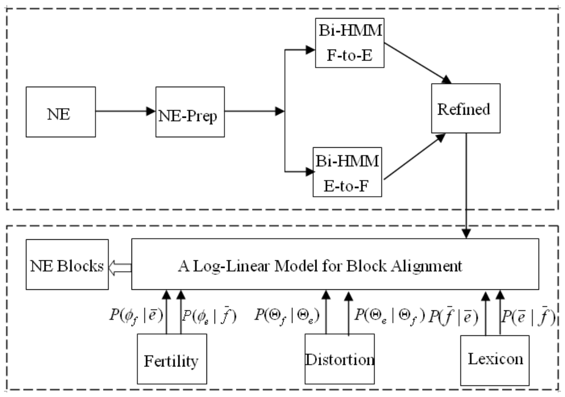
1 Introduction

Cross-lingual natural language applications, such as information retrieval, question answering, and machine translation for web-documents (e.g. Google translation), are becoming increasingly important. However, current state-of-the-art statistical machine translation (SMT) systems cannot yet translate named-entities which are not seen during training. New named-entities, such as person, organization, and location names are continually emerging on the World-Wide-Web. To realize effective cross-lingual natural language applications, handling out-of-vocabulary named-entities is becoming more crucial.

Named entities (NEs) can be translated via transliteration: mapping symbols from one writing system to another. Letters of the source language are typically transformed into the target language with similar pronunciation. Transliteration between languages which share similar alphabets and sound systems is usually not difficult, because the majority of letters remain the same. However, the task is significantly more difficult when the language pairs are considerably different, for example, English-Arabic, English-Chinese, and English-Japanese. In this paper, we focus on *forward* transliteration from Arabic to English.

The work in (Arbabi et al., 1994), to our knowledge, is the first work on machine transliteration of Arabic names into English, French, and Spanish. The idea is to vowelize Arabic names by adding appropriate vowels and utilizing a phonetic look-up table to provide the spelling in the target language. Their framework is strictly applicable within standard Arabic morphological rules. Knight and Graehl (1997) introduced finite state transducers that implement back-transliteration from Japanese to English, which was then extended to Arabic-English in (Stalls and Knight, 1998). Al-Onaizan and Knight (2002) transliterated named entities in Arabic text to English by combining phonetic-based and spelling-based models, and reranking candidates with full-name web counts, named entities co-reference, and contextual web counts. Huang (2005) proposed a specific model for Chinese-English name transliteration with clusterings of names' origins, and appropriate hypotheses are generated given the origins. All of these approaches, however, are not based on a SMT-framework. Technologies developed for SMT are borrowed in Virga and Khudanpur (2003) and AbdulJaleel and Larkey (2003). Standard SMT alignment models (Brown et al., 1993) are used to align letter-pairs within named entity pairs for transliteration. Their approach are generative models for letter-to-letter translations, and the letter-alignment is augmented with heuristics. Letter-level contextual information is shown to be very helpful for transliteration. Oh and Choi (2002) used conversion units for English-Korean Transliteration; Goto et al. (2003) used conversion units, mapping English letter-sequence into Japanese Katakana character string. Li et al. (2004) presented a framework allowing direct orthographical mapping of transliteration units between English and Chinese, and an extended model is presented in Ekbal et al. (2006).

We propose a *block-level* transliteration framework, as shown in Figure 1, to model letter-level context information for transliteration at two levels. First, we propose a bi-stream HMM incorporating letter-clusters to better model the *vowel* and *non-vowel* transliterations with position-information, i.e., *initial* and *final*, to improve the letter-level alignment accuracy. Second, based on the letter-alignment, we propose *letter n-gram* (letter-sequence) alignment models (*block*) to automatically learn the mappings from source letter n-grams to target letter n-grams. A few features specific for transliterations are explored, and a log-linear model is used to combine



NE
NE-Prep
Bi-HMM F-to-E
Bi-HMM E-to-F
Refined
NE Blocks
A Log-Linear Model for Block Alignment
$P(\phi_f \mid \bar{e})$
$P(\phi_e \mid \bar{f})$
$P(\Theta_f \mid \Theta_e)$
$P(\Theta_e \mid \Theta_f)$
$P(\bar{f} \mid \bar{e})$
$P(\bar{e} \mid \bar{f})$
Fertility
Distortion
Lexicon

Figure 1: Transliteration System Structure. The upper-part is the two-directional Bi-Stream HMM for letter-alignment; the lower-part is a log-linear model for combining different feature functions for block-level transliteration.

these features to learn block-level transliteration-pairs from training data. The proposed transliteration framework obtained significant improvements over a strong baseline transliteration approach similar to AbdulJaleel and Larkey (2003) and Virga and Khudanpur (2003).

The remainder of this paper is organized as follows. In Section 2, we formulate the transliteration as a general translation problem; in Section 4, we propose a log-linear alignment model with a local search algorithm to model the letter n-gram translation pairs; in Section 5, experiments are presented. Conclusions and discussions are given in Section 6.

2 Transliteration as Translation

Transliteration can be viewed as a special case of translation. In this approach, source and target NEs are split into letter sequences, and each sequence is treated as a *pseudo sentence*. The appealing reason of formulating transliteration in this way is to utilize advanced alignment models, which share ideas applied also within phrase-based statistical machine translation (Koehn, 2004).

To apply this approach to transliteration, however, some unique aspects should be considered. First, letters should be generated from left to right, without any reordering. Thus, the transliteration models can only execute forward sequential jumps. Second, for unvowelized languages such as Arabic, a single Arabic letter typically maps to less than four English letters. Thus, the fertility for each letter should be recognized to ensure reasonable length relevance. Third, the position of the letter within a NE is important. For example, in Arabic, letters such as "al" at the beginning of the NE can only be translated into "the" or "al". Therefore position information should be considered within the alignment models.

Incorporating the above considerations, transliteration can be formulated as a noisy channel model. Let $f_1^J = f_1 f_2 ... f_J$ denote the source NE with J letters, $e_1^I = e_1 e_2 ... e_I$ be an English transliteration candidate with I letters. According to Bayesian decision rule:

$$\hat{e}_1^I = \arg\max_{\{e_1^I\}} P(e_1^I | f_1^J) = \arg\max_{\{e_1^I\}} P(f_1^J | e_1^I) P(e_1^I), \quad (1)$$

where $P(f_1^J|e_1^I)$ is the *letter translation model* and $P(e_1^I)$ is the English *letter sequence model* corresponding to the monolingual language models in SMT. In this noisy-channel scheme, $P(f_1^J|e_1^I)$ is the key component for transliteration, in which the transliteration between e_1^I and f_1^J can be modeled at either letter-to-letter level, or letter n-gram transliteration level (*block*-level).

Our transliteration models are illustrated in Figure 1. We propose a Bi-Stream HMM of $P(f_1^J|e_1^I)$ to infer letter-to-letter alignments in two directions: Arabic-to-English (F-to-E) and English-to-Arabic (E-to-F), shown in the upper-part in Figure 1; refined alignment is then obtained. We propose a log-linear model to extract block-level transliterations with additional informative features, as illustrated in the lower-part of Figure 1.

3 Bi-Stream HMMs for Transliteration

Standard IBM translation models (Brown et al., 1993) can be used to obtain letter-to-letter translations. However, these models are not directly suitable, because letter-alignment within NEs is strictly left-to-right. This sequential property is well suited to HMMs (Vogel et al., 1996), in which the jumps from the current aligned position can only be forward.

3.1 Bi-Stream HMMs

We propose a bi-stream HMM for letter-alignment within NE pairs. For the source NE f_1^J and a target NE e_1^I, a bi-stream HMM is defined as follows:

$$p(f_1^J|e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} p(f_j|e_{a_j}) p(c_{f_j}|c_{e_{a_j}}) p(a_j|a_{j-1}), \quad (2)$$

where a_j maps f_j to the English letter e_{a_j} at the position a_j in the English named entity. $p(a_j|a_{j-1})$ is the transition probability distribution assuming first-order Markov dependency; $p(f_j|e_{a_j})$ is a letter-to-letter translation lexicon; c_{f_j} is the letter cluster of f_j and $p(c_{f_j}|c_{e_{a_j}})$ is a cluster level translation lexicon. As mentioned in the above, the vowel/non-vowel linguistic features can be utilized to cluster the letters. The letters from the same cluster tend to share the similar letter transliteration forms. $p(c_{f_j}|c_{e_{a_j}})$ enables to leverage such letter-correlation in the transliteration process.

The HMM in Eqn. 2 generates two streams of observations: the letters together with the letters' classes following the distribution of $p(f_j|e_{a_j})$ and $p(c_{f_j}|c_{e_{a_j}})$ at each

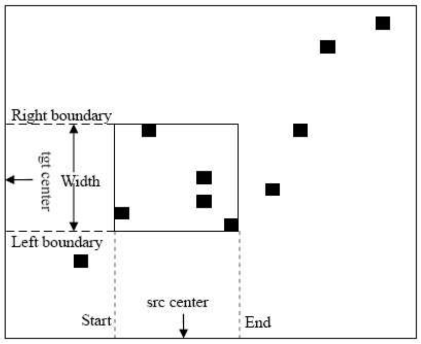


Figure 2: Block of letters for transliteration. A block is defined by the left- and right- boundaries in the NE-pair.

state, respectively. To be in accordance with the monotone nature of the NE's alignment mentioned before, we enforce the following constraints in Eqn. 3, so that the transition can only jump forward or stay at the same state:

$$a_j - a_{j-1} \geq 0 \qquad \forall j \in [1, J]. \tag{3}$$

Since the two streams are conditionally independent given the current state, the extended EM is straightforward, with only small modifications of the standard forward-backward algorithm (Zhao et al., 2005), for parameter estimation.

3.2 Designing Letter-Classes

Pronunciation is typically highly structured. For instance, in English the pronunciation structure of "*cvc*" (*consonant-vowel-consonant*) is common. By incorporating letter classes into the proposed two-stream HMM, the models' expressiveness and robustness can be improved. In this work, we focus on transliteration of Arabic NEs into English. We define six non-overlapping letter classes: *vowel*, *consonant*, *initial*, *final*, *noclass*, and *unknown*. *Initial* and *final* classes represent semantic markers at the beginning or end of NEs such as "Al" and "wAl" (in romanization form). *Noclass* signifies letters which can be pronounced as both a vowel and a consonant depending on context, for example, the English letter "y". The *unknown* class is reserved for punctuations and letters that we do not have enough linguistic clues for mapping them to phonemes.

4 Transliteration Blocks

To further leverage the information from the letter-context beyond the letter-classes incorporated in our bi-stream HMM in Eqn. 2, we define *letter n-grams*, which consist of n *consecutive letters*, as the basic transliteration unit. A *block* is defined as a pair of such letter n-grams which are transliterations of each other. During decoding of unseen NEs, transliteration is performed block-by-block, rather than letter-by-letter. The goal of transliteration model is to learn high-quality transliteration blocks from the training data in a unsupervised fashion.

Specifically, a block X can be represented by its left and right boundaries in the source and target NEs shown in Figure 2:

$$X = (f_j^{j+l}, e_i^{i+k}), \tag{4}$$

where f_j^{j+l} is the source letter-ngram with $(l+1)$ letters in source language, and its projection of $e_i^{\ i+k}$ in the English NE with left boundary at the position of i, and right boundary at $(i+k)$.

We formulate the *block extraction* as a *local search* problem following the work in Zhao and Waibel (2005): given a source letter n-gram f_j^{j+l}, search for the projected boundaries of candidate target letter n-gram $e_i^{\ i+k}$ according to a weighted combination of the diverse features in a *log-linear model* detailed in §4.3. The log-linear model serves as a performance measure to guide the local search, which, in our setup, is *randomized hill-climbing*, to extract bilingual letter n-gram transliteration pairs.

4.1 Features for Block Transliteration

Three features: *fertility*, *distortion*, and *lexical translation* are investigated for inferring transliteration blocks from the NE pairs. Each feature corresponds to one aspect of the block within the context of a given NE pair.

4.1.1 Letter n-gram Fertility

The *fertility* $P(\phi|e)$ of a target letter e specifies the probability of generating ϕ source letters for transliteration. The fertilities can be easily read-off from the letter-alignment, i.e., the output from the Bi-stream HMM. Given letter fertility model $P(\phi|e_i)$, a target letter n-gram e_1^I, and a source n-gram f_1^J of length J, we compute a probability of *letter n-gram length* relevance: $P(J|e_1^I)$ via a dynamic programming.

The probability of generating J letters by the English letter n-gram e_1^I is defined:

$$P(J|e_1^I) = \max_{\{\phi_1^I, J=\sum_{i=1}^I \phi_i\}} \prod_{i=1}^{I} P(\phi_i|e_i). \tag{5}$$

The recursively updated cost $\phi[j,i]$ in dynamic programming is defined as follows:

$$\phi[j,i] = max \begin{cases} \phi[j,i-1] + \log P_{Null}(0|e_i) \\ \phi[j-1,i-1] + \log P_\phi(1|e_i) \\ \phi[j-2,i-1] + \log P_\phi(2|e_i) \\ \phi[j-3,i-1] + \log P_\phi(3|e_i) \end{cases}, \tag{6}$$

where $P_{Null}(0|e_i)$ is the probability of generating a Null letter from e_i; $P_\phi(k{=}1|e_i)$ is the letter-fertility model of generating *one* source letter from e_i; $\phi[j,i]$ is the cost so far for generating j letters from i consecutive English letters (letter n-gram) $e_1^i : e_1, \cdots, e_i$.

After computing the cost of $\phi[J, I]$, the probability $P(J|e_1^I)$ is computed for generating the length of the source NE f_1^J from the English NE e_1^I shown in Eqn. 5. With this letter n-gram fertility model, for every block, we can compute a fertility score to estimate how relevant the lengths of the transliteration-pairs are.

4.1.2 Distortion of Centers

When aligning blocks of letters within transliteration pairs, we expect most of them are close to the diagonal due to the monotone alignment nature. Thus, a simple position metric is proposed for each block considering the relative positions within NE-pairs.

The center $\odot_{f_j^{j+l}}$ of the source phrase f_j^{j+l} with a length of $(l+1)$ is simply a normalized relative position in the source entity defined as follows:

$$\odot_{f_j^{j+l}} = \frac{1}{l+1} \sum_{j'=j}^{j'=j+l} \frac{j'}{l+1}. \tag{7}$$

For the center of English letter-phrase e_i^{i+k}, we first define the expected corresponding relative center for every source letter $f_{j'}$ using the lexicalized position score as follows:

$$\odot_{e_i^{i+k}}(f_{j'}) = \frac{1}{k+1} \cdot \frac{\sum_{i'=i}^{(i+k)} i' \cdot P(f_{j'}|e_{i'})}{\sum_{i'=i}^{(i+k)} P(f_{j'}|e_{i'})}, \tag{8}$$

where $P(f_{j'}|e_i)$ is the letter translation lexicon estimated in IBM Models 1$\sim$5. i is the position index, which is weighted by the letter-level translation probabilities; the term of $\sum_{i'=i}^{i+k} P(f_{j'}|e_{i'})$ provides a normalization so that the expected center is within the range of the target length. The expected center for e_i^{i+k} is simply the average of the $\odot_{e_i^{i+k}}(f_{j'})$:

$$\odot_{e_i^{i+k}} = \frac{1}{l+1} \sum_{j'=j}^{j+l} \odot_{e_i^{i+k}}(f_{j'}) \tag{9}$$

Given the estimated centers of $\odot_{f_j^{j+l}}$ and $\odot_{e_i^{i+k}}$, we can compute how close they are via the probability of $P(\odot_{f_j^{j+l}}|\odot_{e_i^{i+k}})$. In our case, because of the monotone alignment nature of transliteration pairs, a simple gaussian model is employed to enforce that the point $(\odot_{e_i^{i+k}}, \odot_{f_j^{j+l}})$ is not far away from the diagonal.

4.1.3 Letter Lexical Transliteration

Similar to IBM Model-1 (Brown et al., 1993), we use a "bag-of-letter" generative model within a block to approximate the lexical transliteration equivalence:

$$P(f_j^{j+l}|e_i^{i+k}) = \prod_{j'=j}^{j+l} \sum_{i'=i}^{i+k} P(f_{j'}|e_{i'}) P(e_{i'}|e_i^{i+k}), \tag{10}$$

where $P(e_{i'}|e_i^{i+k}) \simeq 1/(k+1)$ is approximated by a bag-of-word unigram. Since named entities are usually relatively short, this approximation works reasonably well in practice.

4.2 Extended Feature Functions

Because of the underlying nature of the noisy-channel model in our proposed transliteration approach in Section 2, the three base feature functions are extended to cover the directions both from target-to-source and source-to-target. Therefore, we have in total six feature functions for inferring transliteration blocks from a named entity pair.

Besides the above six feature functions, we also compute the average letter-alignment links per block. We count the number of letter-alignment links within the block, and normalize the number by the length of the source letter-ngram. Note that, we can refine the letter-alignment by growing the intersections of the two direction letter-alignments from Bi-stream HMM via additional aligned letter-pairs seen in the union of the two. In a way, this approach is similar to those of refining the word-level alignment for SMT in (Och and Ney, 2003). This step is shown in the upper-part in Figure 1.

Overall, our proposed feature functions cover relatively different aspects for transliteration blocks: the block level length relevance probability in Eqn. 5, lexical translation equivalence, and positions' distortion from a gaussian distribution in Eqn. 8, in both directions; and the average number of letter-alignment links within the block. Also, these feature functions are positive and bounded within $[0, 1]$. Therefore, it is suitable to apply a log-linear model (in §4.3) to combine the *weighted* individual strengths from the proposed feature functions for better modeling the quality of the candidate transliteration blocks. This log-linear model will serve as a performance measure in a local-search in §4.4 for inferring transliteration blocks.

4.3 Log-Linear Transliteration Model

We propose a log-linear model to combine the seven feature functions in §4.1 with proper weights as in Eqn. 11:

$$Pr(X|\mathbf{e}, \mathbf{f}) = \frac{\exp(\sum_{m=1}^{M} \lambda_m \phi_m(X, \mathbf{e}, \mathbf{f}))}{\sum_{\{X'\}} \exp(\sum_{m=1}^{M} \lambda_m \phi_m(X', \mathbf{e}, \mathbf{f}))}, \tag{11}$$

where $\phi_m(X, \mathbf{e}, \mathbf{f})$ are the real-valued bounded feature functions corresponding to the seven models introduced in §4.1. The log-linear model's parameters are the weights $\{\lambda_m\}$ associated with each feature function.

With hand-labeled data, $\{\lambda_m\}$ can be learnt via generalized iterative scaling algorithm (GIS) (Darroch and Ratcliff, 1972) or improved iterative scaling (IIS) (Berger et al., 1996). However, as these algorithms are computationally expensive, we apply an alternative approach using a simplex down-hill algorithm to optimize the weights toward better F-measure of block transliterations. Each feature function corresponds to one dimension in the simplex, and the local optimum only happens at a vertex of the simplex. Simplex-downhill has several advantages: it is an efficient approach for optimizing multi-variables given some performance measure. We compute the F-measure against a gold-standard block set extracted from hand-labeled letter-alignment.

To build gold-standard blocks from hand-labeled letter-alignment, we propose the *block transliteration coherence* in a two-stage fashion. First is the forward projection: for each candidate source letter-ngram f_j^{j+n}, search for its *left-most* e_l and *right-most* e_r projected positions in the *target* NE according to the given letter-alignment. Second is the backward projection: for the target letter-gram e_l^r, search for its *left-most* $f_{l'}$ and *right-most* $f_{r'}$ projected positions in the *source* NE. Now if $l' \geq j$ and $r' \leq j+n$, i.e. f_l^r is contained within the source letter-ngram f_j^{j+n}, then this block $X = (f_j^{j+n}, e_l^r)$ is defined as *coherent* for the aligned pairs: (f_j^{j+n}, e_l^r). We accept coherent X as gold-standard blocks. This block transliteration coherence is generally sound for extracting the gold-blocks mostly because of the the monotone left-to-right nature of the letter-alignment for transliteration. A related coherence assumption can be found in (Fox, 2002), where their assumption on phrase-pairs for statistical machine translation is shown to be somewhat restrictive for SMT. This is mainly because the word alignment is often *non-monotone*, especially for langauge-pairs from different families such as Arabic-English and Chinese-English.

4.4 Aligning Letter-Blocks: a Local Search

Aligning the blocks within NE pairs can be formulated as a local search given the heuristic function defined in Eqn. 11. To be more specific: given a Arabic letter-ngram f_j^{j+l}, our algorithm searches for the best translation candidate e_i^{i+k} in the target named entities. In our implementation, we use stochastic hill-climbing with Eqn. 11 as the performance measure. Down-hill moves are accepted to allow one or two left and right null letters to be attached to e_i^{i+k} to expand the table of transliteration-blocks.

To make the local search more effective, we normalize the letter translation lexicon $p(f|e)$ within the parallel entity pair as in:

$$\hat{P}(f|e) = \frac{P(f|e)}{\sum_{j'=1}^{J} P(f_{j'}|e)}. \tag{12}$$

In this way, the distribution of $\hat{P}(f|e)$ is sharper and more focused in the context of an entity pair.

Overall, given the parallel NE pairs, we can train the letter level translation models in both directions via the Bi-stream HMM in Eqn. 2. From the letter-alignment, we can build the letter translation lexicons and fertility tables. With these tables, the base feature functions are then computed for each candidate block, and the features are combined in the log-linear model in Eqn. 11. Given a named-entity pair in the training data, we rank all the transliteration blocks by the scores using the log-linear model. This step is shown in the lower-part in Figure 1.

4.5 Decoding Unseen NEs

The decoding of NEs is an extension to the noisy-channel scheme in Eqn. 1. In our configurations for NE transliteration, the extracted transliteration blocks are used. Our letter ngram is a standard letter-ngram model trained using the SriLM toolkit (Stolcke, 2002). To transliterate the unseen NEs, the decoder (Hewavitharana et al., 2005) is configured for monotone decoding. It loads the transliteration blocks and the letter-ngram LM, and it decodes the unseen Arabic named entities with block-based transliteration from left to right.

5 Experiments

5.1 The Data

We have 74,887 bilingual geographic names from LDC2005G01-NGA, 11,212 bilingual person names from LDC2005G02[1], and about 6,000 bilingual names extracted from the BAMA[2] dictionary. In total, there are 92,099 NE pairs. We split them into three parts: 91,459 pairs as the training dataset, 100 pairs as the development dataset, and 540 unique NE pairs as the held-out dataset.

An additional test set is collected from the TIDES 2003 Arabic-English machine translation evaluation test set. The 663 sentences contain 286 unique words, which were not covered by the available training data. From this set of untranslated words, we manually labeled the entities of persons, locations and organizations, giving a total of 97 unique un-translated NEs. The BAMA toolkit was used to romanize the Arabic words. Some names from this test set are shown in Figure 1.

These untranslated NEs make up only a very small fraction of all words in the test set. Therefore, having correct transliterations would give only small improvements in terms of BLEU (Papineni et al., 2002) and NIST scores. However, successfully translating these unknown NEs is very crucial for cross-lingual distillation tasks or question-answering based on the MT-output.

[1]The corpus is provided as FOUO (for official use only) in the *DARPA-GALE* project

[2]LDC2004L02: Buckwalter Arabic Morphological Analyzer version 2.0

Table 1: Test Set Examples.

Arabic	BAMA	Reference
غرابو	grAbw	Grabo
قشطة	qXTp	Qishta
فايتساخر	fAytsAxr	Weizsacker
والدهماني	wAldHmAny	al-Dahmani
زيلويغير	zylwygyr	Zellweger
ثاكسين	vAksyn	Thaksin

To evaluate the transliteration performance, we use *edit-distance* between the hypothesis against a reference set. This is to count the number of insertions, deletions, and substitutions required to correct the hypothesis to match the given reference. An edit-distance of zero is a perfect match. However, NEs typically have more than one correct variant. For example, the Arabic name "mHmd" (in romanized form) can be transliterated as Muhammad or Mohammed; both are considered as correct transliterations. Ideally, we want to have all variants as reference transliterations. To enable our transliteration evaluation to be more informative given only one reference, edit-distance of one between hypothesis and reference is considered to be an acceptable match.

5.2 Comparison of Transliteration Models

We compare the performance of three systems within our proposed framework in Figure.1: the baseline Block system, a system in which we use a log-linear combination of alignment features as described in §4.3, we call the the L-Block system, and finally a system, which also uses the bi-stream HMM alignment model as described in §3. This last system will be denoted LCBE system.

The baseline is based on the refined letter-alignment from the two directions of IBM-Model-4, trained with a scheme of $1^5h^54^5$ using GIZA++ (Och and Ney, 2004). The final alignment was obtained by growing the intersections between Arabic-to-English (AE) and English-to-Arabic (EA) alignments with additional aligned letter-pairs seen in the union. This is to compensate for the inherent asymmetry in alignment models. Blocks (letter-ngram pairs) were collected directly from the refined letter-alignment, using the same algorithm as described in §4.3 for extracting gold-standard letter blocks. There is no length restrictions to the letter-ngram extracted in our system. All the blocks were then scored using relative frequencies and lexical scores in both directions, similar to the scoring of phrase-pairs in SMT (Koehn, 2004).

In the L-Block system additional feature functions as defined in §4.1 were computed on top of the letter-level alignment obtained from the baseline system. A log-linear model combining these features was learned with the gold-blocks described in §4.3. Transliteration blocks were extracted using the local-search §4.4. The other components remained the same as in the baseline system.

Table 2: Transliteration accuracy for different transliteration models.

System	Accuracy
Baseline	39.18%
L-Block	41.24%
LCBE	46.39%

The LCBE system is an extension to both the baseline and the L-Block system. The key difference in LCBE is that our proposed bi-stream HMM in Eqn. 2 was applied in both directions with extended letter-classes. The resulting combined alignment was used together with all features of the L-Block system to guide the local-search for extracting the blocks. The same procedure of decoding was then carried out for the unseen NEs using the extracted blocks.

To build the letter language model for the decoding process, we first split the English entities into characters; additional *position indicators* "_begin" and "_end" were added to the begin and end position of the named-entity; "_middle" was added between the first name and last name. A letter-trigram language model with SRI LM toolkit (Stolcke, 2002) was then built using the target side (English) of NE pairs tagged with the above position information.

Table 2 shows that the baseline system gives an accuracy of 39.18%, while the extended systems L-Block and LCBE give 41.24% and 46.39%, respectively. These results show that the additional features besides the letter-alignment are helpful. The L-Block system, which uses these features, outperforms the baseline system significantly by 2.1% absolute in accuracy. The results also show that the bi-stream HMM alignment, which uses not only the letters but also the letter-classes, leads to significant improvement. It outperforms the L-Block system, which does not leverage the letter-classes and monotone alignment, by 4.15% absolute.

5.3 Incorporation of Spell Checking

Our spelling-checker is based on the suggested word-forms from web search engines for ambiguous candidates. We collected web statistics frequency for both the proposed transliteration candidates from our system, and also the suggested candidates from web-search engines. All the candidates were re-ranked by their frequencies.

Figure 3 shows the performances on the held-out set, using system LCBE augmented with a spell-checker (*LCBE+Spell*), with varying sizes of N-best hypotheses lists. The held-out set contains 540 unique named entity pairs. We show accuracy when exact match is requested and when an edit distances of one is allowed.

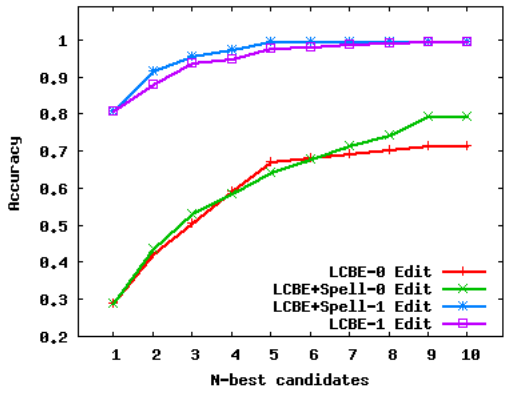


Figure 3: Transliteration accuracy of LCBE and LCBE+Spell models for 540 named entity pairs in the held-out set.

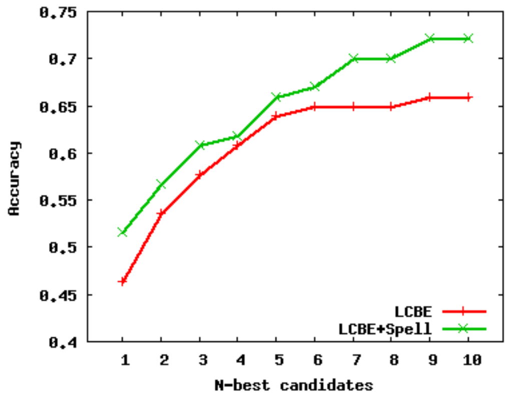


Figure 4: Transliteration accuracy of N-best hypotheses for LCBE and LCBE+Spell models it the MT-03 test set.

Figure 4 shows the performances in the unseen test set of LCBE and LCBE+Spell, with varying sizes of N-best hypotheses lists. LCBE+Spell reaches 52% accuracy in 1-best hypothesis. In the 5-best and 10-best cases, the accuracies of LCBE+Spell system archive the highest performances with 66% and 72.16% respectively. The spell-checker increases the 1-best accuracy by 11.12% and the 10-best accuracy by 7.69%. All these improvements are statistically significant. These results are also comparable to other state-of-the-art statistical Arabic name transliteration systems such as (Al-Onaizan and Knight, 2002).

5.4 Comparison with the Google Web Translation

We finally compared our best system with the state-of-the-art Arabic-English Google Web Translation (Google). Table 3 shows transliteration examples from our best system in comparison with Google (as in June 20, 2006)[3]. The Google system achieved 45.36% accuracy for the 1-best hypothesis, which is comparable to the results when using the LCBE transliteration system, while LCBE+Spell archived 52%.

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

Table 3: Transliteration examples between LCBE+Spell and Google web translation.

Source	Reference	LCBE+Spell	Google
سومای	Sumaye	Sumaye	Somai
هازوميتسو	Hazumitsu	Hazumitsu	Hazoumitso
يلاه	Yalahow	Ylahu	Elaho
نكباخت	Nikbakht	Nkbakht	Nkbacht
ميكوياس	Mikulas	Mikulas	Mikoias
كوماراتونج	Kumaratunga	Kumaratunga	Kumaratung
همدان	Hamdan	Hamdan	Hamedan
لمازانداران	Mazandaran	Mazandaran	Mazandaran
ويكر مسينغه	Wickremasinghe	Wikramsinghe	The Ekermsingh

6 Conclusions and Discussions

In this paper we proposed a novel transliteration model. Viewing transliteration as a translation task we adopt alignment and decoding techniques used in a phrase-based statistical machine translation system to work on letter sequences instead of word sequences. To improve the performance we extended the HMM alignment model into a bi-stream HMM alignment by incorporating letter-classes into the alignment process. We also showed that a block-extraction approach, which uses a log-linear combination of multiple alignment features, can give significant improvements in transliteration accuracy. Finally, spell-checking based on work occurrence statistics obtained from the web gave an additional boost in transliteration accuracy.

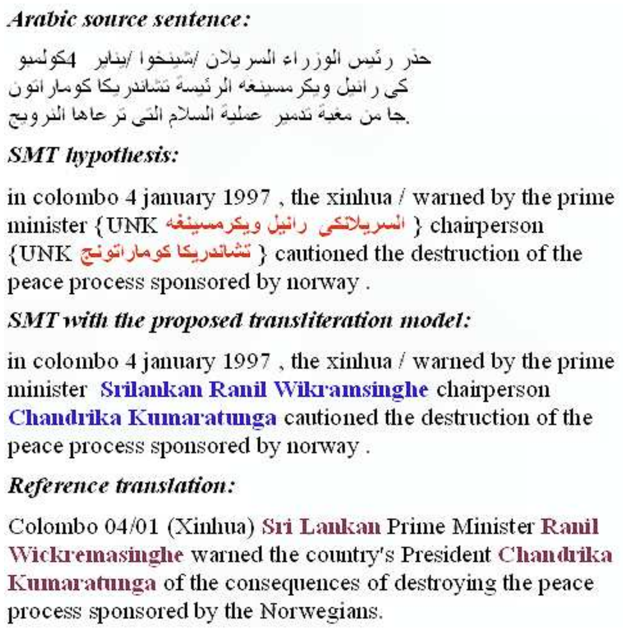
The goal for this work is to improve the quality of machine translation, esp. when used in cross-lingual information retrieval and distillation tasks, by incorporating the proposed framework to handle unknown words. Figure 5 gives an example of the difference named entity transliteration can make. Shown are the original SMT system output, the translation when the proposed transliteration models are used to translate the unknown named-entities, and the reference translation. A comparison of the two SMT outputs indicates that integrating the proposed transliteration model into our machine translation system can significantly improve translation utility.

Acknowledgment

This work was partially supported by grants from DARPA (GALE project) and NFS (Str-Dust project).

References

Nasreen AbdulJaleel and Leah Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, New Orleans, LA, USA, November.


Arabic source sentence:

حذر رئيس الوزراء السريلان /شينخوا /يناير 4كولمبو
كي رانيل ويكرمسينغه الرئيسة تشاندريكا كوماراتون
جا من مغبة تدمير عملية السلام التي ترعاها النرويج.

SMT hypothesis:

in colombo 4 january 1997 , the xinhua / warned by the prime minister {UNK السريلانكي رانيل ويكرمسينغه } chairperson {UNK تشاندريكا كوماراتونج } cautioned the destruction of the peace process sponsored by norway .

SMT with the proposed transliteration model:

in colombo 4 january 1997 , the xinhua / warned by the prime minister **Srilankan Ranil Wikramsinghe** chairperson **Chandrika Kumaratunga** cautioned the destruction of the peace process sponsored by norway .

Reference translation:

Colombo 04/01 (Xinhua) **Sri Lankan** Prime Minister **Ranil Wickremasinghe** warned the country's President **Chandrika Kumaratunga** of the consequences of destroying the peace process sponsored by the Norwegians.

Figure 5: Incorporation of the transliteration model to our SMT System.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of ACL Workshop on Computational Approaches to Semitic Languages*, Philadelphia, PA, USA.

Mansur Arbabi, Scott M. Fischthal, Vincent C. Cheng, and Elizabeth Bart. 1994. Algorithms for Arabic name transliteration. In *IBM Journal of Research and Development*, volume 38(2), pages 183–193.

Adam L. Berger, Vincent Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Computational Linguistics*, volume 22 of *1*, pages 39–71, March.

Peter 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.

J.N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. In *Annals of Mathematical Statistics*, volume 43, pages 1470–1480.

Asif Ekbal, S. Naskar, and S. Bandyopadhyay. 2006. A modified joint source channel model for machine transliteration. In *Proceedings of COLING/ACL*, pages 191–198, Australia.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 304–311, Philadelphia, PA, July 6-7.

Isao Goto, Naoto Kato, Noriyoshi Uratani, and Terumasa Ehara. 2003. Transliteration considering context information based on the maximum entropy method. In *Proceedings of MT-Summit IX*, New Orleans, Louisiana, USA.

Sanjika Hewavitharana, Bing Zhao, Almut Silja Hildebrand, Matthias Eck, Chiori Hori, Stephan Vogel, and Alex Waibel. 2005. The CMU statistical machine translation system for IWSLT2005. In *The 2005 International Workshop on Spoken Language Translation*.

Fei Huang. 2005. Cluster-specific name transliteration. In *Proceedings of the HLT-EMNLP 2005*, Vancouver, BC, Canada, October.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, Madrid, Spain.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based smt. In *Proceedings of the Conference of the Association for Machine Translation in the Americans (AMTA)*, Washington DC, USA.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of 42nd ACL*, pages 159–166, Barcelona, Spain.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 1:29, pages 19–51.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. In *Computational Linguistics*, volume 30, pages 417–449.

Jong-Hoon Oh and Key-Sun Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Proceedings of COLING-2002*, pages 1–7, Taipei, Taiwan.

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 Association for Computational Linguistics (ACL 02)*, pages 311–318, Philadelphia, PA, July.

Bonnie Stalls and Kevin Knight. 1998. Translating names and technical terms in Arabic text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*, Montreal, Quebec, Canada.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904, Denver.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL Workshop on Multi-lingual Named Entity Recognition*, Edmonton, Canada.

Stephan. Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM based word alignment in statistical machine translation. In *Proc. The 16th Int. Conf. on Computational Lingustics, (COLING-1996)*, pages 836–841, Copenhagen, Denmark.

Bing Zhao and Alex Waibel. 2005. Learning a log-linear model with bilingual phrase-pair features for statistical machine translation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, Jeju Island, Korean, October.

Bing Zhao, Eric P. Xing, and Alex Waibel. 2005. Bilingual word spectral clustering for statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 25–32, Ann Arbor, Michigan, June.

Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion

Sittichai Jiampojamarn, Grzegorz Kondrak and Tarek Sherif
Department of Computing Science,
University of Alberta,
Edmonton, AB, T6G 2E8, Canada
{sj,kondrak,tarek}@cs.ualberta.ca

Abstract

Letter-to-phoneme conversion generally requires aligned training data of letters and phonemes. Typically, the alignments are limited to one-to-one alignments. We present a novel technique of training with many-to-many alignments. A letter chunking bigram prediction manages double letters and double phonemes automatically as opposed to preprocessing with fixed lists. We also apply an HMM method in conjunction with a local classification model to predict a global phoneme sequence given a word. The many-to-many alignments result in significant improvements over the traditional one-to-one approach. Our system achieves state-of-the-art performance on several languages and data sets.

1 Introduction

Letter-to-phoneme (L2P) conversion requires a system to produce phonemes that correspond to a given written word. Phonemes are abstract representations of how words should be pronounced in natural speech, while letters or graphemes are representations of words in written language. For example, the phonemes for the word *phoenix* are [f i n ɪ k s].

The L2P task is a crucial part of speech synthesis systems, as converting input text (graphemes) into phonemes is the first step in representing sounds. L2P conversion can also help improve performance in spelling correction (Toutanova and Moore, 2001). Unfortunately, proper nouns and unseen words prevent a table look-up approach. It is infeasible to construct a lexical database that includes every word in the written language. Likewise, orthographic complexity of many languages prevents us from using hand-designed conversion rules. There are always exceptional rules that need to be added to cover a large vocabulary set. Thus, an automatic L2P system is desirable.

Many data-driven techniques have been proposed for letter-to-phoneme conversion systems, including pronunciation by analogy (Marchand and Damper, 2000), constraint satisfaction (Van Den Bosch and Canisius, 2006), Hidden Markov Model (Taylor, 2005), decision trees (Black et al., 1998), and neural networks (Sejnowski and Rosenberg, 1987). The training data usually consists of written words and their corresponding phonemes, which are not aligned; there is no explicit information indicating individual letter and phoneme relationships. These relationships must be postulated before a prediction model can be trained.

Previous work has generally assumed one-to-one alignment for simplicity (Daelemans and Bosch, 1997; Black et al., 1998; Damper et al., 2005). An expectation maximization (EM) based algorithm (Dempster et al., 1977) is applied to train the aligners. However, there are several problems with this approach. Letter strings and phoneme strings are not typically the same length, so null phonemes and null letters must be introduced to make one-to-one-alignments possible, Furthermore, two letters frequently combine to produce a single phoneme (double letters), and a single letter can sometimes produce two phonemes (double phonemes).

To help address these problems, we propose an automatic many-to-many aligner and incorporate it into a generic classification predictor for letter-to-phoneme conversion. Our many-to-many aligner automatically discovers double phonemes and double letters, as opposed to manually preprocessing data by merging phonemes using fixed lists. To our knowledge, applying many-to-many alignments to letter-to-phoneme conversion is novel.

Once we have our many-to-many alignments, we use that data to train a prediction model. Many phoneme prediction systems are based on local prediction methods, which focus on predicting an individual phoneme given each letter in a word. Conversely, a method like pronunciation by analogy (PbA) (Marchand and Damper, 2000) is considered a global prediction method: predicted phoneme sequences are considered as a whole. Recently, Van Den Bosch and Canisius (2006) proposed trigram class prediction, which incorporates a constraint satisfaction method to produce a global prediction for letter-to-phoneme conversion. Both PbA and trigram class prediction show improvement over predicting individual phonemes, confirming that L2P systems can benefit from incorporating the relationship between phonemes in a sequence.

In order to capitalize on the information found in phoneme sequences, we propose to apply an HMM method after a local phoneme prediction process. Given a candidate list of two or more possible phonemes, as produced by the local predictor, the HMM will find the best phoneme sequence. Using this approach, our system demonstrates an improvement on several language data sets.

The rest of the paper is structured as follows. We describe the letter-phoneme alignment methods including a standard one-to-one alignment method and our many-to-many approach in Section 2. The alignment methods are used to align graphemes and phonemes before the phoneme prediction models can be trained from the training examples. In Section 3, we present a letter chunk prediction method that automatically discovers double letters in grapheme sequences. It incorporates our many-to-many alignments with prediction models. In Section 4, we present our application of an HMM method to the local prediction results. The results of experiments on several language data sets are discussed in Section 5. We conclude and propose future work in Section 6.

2 Letter-phoneme alignment

2.1 One-to-one alignment

There are two main problems with one-to-one alignments:

1. Double letters: two letters map to one phoneme (e.g. *sh* - [ʃ], *ph* - [f]).

2. Double phonemes: one letter maps to two phonemes (e.g. *x* - [k s], *u* - [j u]).

First, consider the double letter problem. In most cases when the grapheme sequence is longer than the phoneme sequence, it is because some letters are silent. For example, in the word *abode*, pronounced [ə b o d], the letter e produces a null phoneme (ϵ). This is well captured by one-to-one aligners. However, the longer grapheme sequence can also be generated by double letters; for example, in the word *king*, pronounced [k ɪ ŋ], the letters *ng* together produce the phoneme [ŋ]. In this case, one-to-one aligners using null phonemes will produce an incorrect alignment. This can cause problems for the phoneme prediction model by training it to produce a null phoneme from either of the letters n or g.

In the double phoneme case, a new phoneme is introduced to represent a combination of two (or more) phonemes. For example, in the word *fume* with phoneme sequence [f j u m], the letter u produces both the [j] and [u] phonemes. There are two possible solutions for constructing a one-to-one alignment in this case. The first is to create a new phoneme by merging the phonemes [j] and [u]. This requires constructing a fixed list of new phonemes before beginning the alignment process. The second solution is to add a null letter in the grapheme sequence. However, the null letter not only confuses the phoneme prediction model, but also complicates the the phoneme generation phase.

For comparison with our many-to-many approach, we implement a one-to-one aligner based on the epsilon scattering method (Black et al., 1998). The method applies the EM algorithm to estimate

Algorithm 1: Pseudocode for a many-to-many expectation-maximization algorithm.

```
Algorithm:EM-many2many
Input: $x^T, y^V, maxX, maxY$
Output: $\gamma$
forall mapping operations $z$ do
    $\gamma(z) := 0$
foreach sequence pair $(x^T, y^V)$ do
    Expectation-many2many($x^T, y^V, maxX, maxY, \gamma$)
Maximization-Step($\gamma$)
```

the probability of mapping a letter l to a phoneme p, $P(l, p)$. The initial probability table starts by mapping all possible alignments between letters and phonemes for each word in the training data, introducing all possible null phoneme positions. For example, the word/phoneme-sequence pair *abode* [ə b o d] has five possible positions where a null phoneme can be added to make an alignment.

The training process uses the initial probability table $P(l, p)$ to find the best possible alignments for each word using the Dynamic Time Warping (DTW) algorithm (Sankoff and Kruskal, 1999). At each iteration, the probability table $P(l, p)$ is re-calculated based on the best alignments found in that iteration. Finding the best alignments and re-calculating the probability table continues iteratively until there is no change in the probability table. The final probability table $P(l, p)$ is used to find one-to-one alignments given graphemes and phonemes.

2.2 Many-to-Many alignment

We present a many-to-many alignment algorithm that overcomes the limitations of one-to-one aligners. The training of the many-to-many aligner is an extension of the forward-backward training of a one-to-one stochastic transducer presented in (Ristad and Yianilos, 1998). Partial counts are counts of all possible mappings from letters to phonemes that are collected in the γ table, while mapping probabilities (initially uniform) are maintained in the δ table. For each grapheme-/phoneme-sequence pair (x, y), the *EM-many2many* function (Algorithm 1) calls the *Expectation-many2many* function (Algorithm 2) to collect partial counts. T and V are the lengths of x and y respectively. The $maxX$ and $maxY$ variables are the maximum lengths of subsequences used in a single mapping operation for x and y. (For the task at hand, we set both $maxX$ and $maxY$ to 2.) The *Maximization-step* function simply normalizes the partial counts to create a probability distribution. Normalization can be done over the whole table to create a joint distribution or per grapheme to create a conditional distribution.

Algorithm 2: Pseudocode for a many-to-many expectation algorithm.

```
Algorithm:Expectation-many2many
Input: $x^T, y^V, maxX, maxY, \gamma$
Output: $\gamma$
$\alpha :=$ Forward-many2many $(x^T, y^V, maxX, maxY)$
$\beta :=$ Backward-many2many $(x^T, y^V, maxX, maxY)$
if $(\alpha_{T,V} = 0)$ then
    return
for $t = 0...T$ do
    for $v = 0...V$ do
        if $(t > 0 \wedge DELX)$ then
            for $i = 1...maxX$ st $t - i \geq 0$ do
                $\gamma(x_{t-i+1}^t, \epsilon) += \frac{\alpha_{t-i,v}\delta(x_{t-i+1}^t, \epsilon)\beta_{t,v}}{\alpha_{T,V}}$
        if $(v > 0 \wedge DELY)$ then
            for $j = 1...maxY$ st $v - j \geq 0$ do
                $\gamma(\epsilon, y_{v-j+1}^v) += \frac{\alpha_{t,v-j}\delta(\epsilon, y_{v-j+1}^v)\beta_{t,v}}{\alpha_{T,V}}$
        if $(v > 0 \wedge t > 0)$ then
            for $i = 1...maxX$ st $t - i \geq 0$ do
                for $j = 1...maxY$ st $v - j \geq 0$ do
                    $\gamma(x_{t-i+1}^t, y_{v-j+1}^v) += \frac{\alpha_{t-i,v-j}\delta(x_{t-i+1}^t, y_{v-j+1}^v)\beta_{t,v}}{\alpha_{T,V}}$
```

The *Forward-many2many* function (Algorithm 3) fills in the table α, with each entry $\alpha(t, v)$ being the sum of all paths through the transducer that generate the sequence pair (x_1^t, y_1^v). Analogously, the *Backward-many2many* function fills in β, with each entry $\beta(t, v)$ being the sum of all paths through the transducer that generate the sequence pair (x_t^T, y_v^V). The constants $DELX$ and $DELY$ indicate whether or not deletions are allowed on either side. In our system, we allow letter deletions (i.e. mapping of letters to null phoneme), but not phoneme deletions.

Expectation-many2many first calls the two functions to fill the α and β tables, and then uses the probabilities to calculate partial counts for every possible mapping in the sequence pair. The partial count collected at positions t and v in the sequence pair is the sum of all paths that generate the sequence pair and go through (t, v), divided by the sum of all paths that generate the entire sequence pair ($\alpha(T, V)$).

Once the probabilities are learned, the Viterbi

Algorithm 3: Pseudocode for a many-to-many forward algorithm.

```
Algorithm: Forward-many2many
Input: $(x^T, y^V, maxX, maxY)$
Output: $\alpha$
$\alpha_{0,0} := 1$
for $t = 0...T$ do
    for $v = 0...V$ do
        if $(t > 0 \vee v > 0)$ then
            $\alpha_{t,v} = 0$
        if $(t > 0 \wedge DELX)$ then
            for $i = 1...maxX$ st $t - i \geq 0$ do
                $\alpha_{t,v} += \delta(x_{t-i+1}^t, \epsilon)\alpha_{t-i,v}$
        if $(v > 0 \wedge DELY)$ then
            for $j = 1...maxY$ st $v - j \geq 0$ do
                $\alpha_{t,v} += \delta(\epsilon, y_{v-j+1}^v)\alpha_{t,v-j}$
        if $(v > 0 \wedge t > 0)$ then
            for $i = 1...maxX$ st $t - i \geq 0$ do
                for $j = 1...maxY$ st $v - j \geq 0$ do
                    $\alpha_{t,v} += \delta(x_{t-i+1}^t, y_{v-j+1}^v)\alpha_{t-i,v-j}$
```

algorithm can be used to produce the most likely alignment as in the following equations. Back pointers to maximizing arguments are kept at each step so the alignment can be reconstructed.

$$\alpha(0,0) = 1 \tag{1}$$

$$\alpha(t,v) = \max_{\substack{1 \leq i \leq maxX, \\ 1 \leq j \leq maxY}} \begin{cases} \delta(x_{t-i+1}^t, \epsilon)\alpha_{t-i,v} \\ \delta(\epsilon, y_{v-j+1}^v)\alpha_{t,v-j} \\ \delta(x_{t-i+1}^t, y_{v-j+1}^v)\alpha_{t-i,v-j} \end{cases} \tag{2}$$

Given a set of words and their phonemes, alignments are made across graphemes and phonemes. For example, the word *phoenix*, with phonemes [f i n ɪ k s], is aligned as:

ph	*oe*	*n*	*i*	*x*
|	|	|	|	|
f	i	n	ɪ	ks

The letters *ph* are an example of the double letter problem (mapping to the single phoneme [f]), while the letter *x* is an example of the double phoneme problem (mapping to both [k] and [s] in the phoneme sequence). These alignments provide more accurate grapheme-to-phoneme relationships for a phoneme prediction model.

3 Letter chunking

Our new alignment scheme provides more accurate alignments, but it is also more complex — sometimes a prediction model should predict two phonemes for a single letter, while at other times the prediction model should make a prediction based on a pair of letters. In order to distinguish between these two cases, we propose a method called "letter chunking".

Once many-to-many alignments are built across graphemes and phonemes, each word contains a set of letter chunks, each consisting of one or two letters aligned with phonemes. Each letter chunk can be considered as a grapheme unit that contains either one or two letters. In the same way, each phoneme chunk can be considered as a phoneme unit consisting of one or two phonemes. Note that the double letters and double phonemes are implicitly discovered by the alignments of graphemes and phonemes. They are not necessarily consistent over the training data but based on the alignments found in each word.

In the phoneme generation phase, the system has only graphemes available to predict phonemes, so there is no information about letter chunk boundaries. We cannot simply merge any two letters that have appeared as a letter chunk in the training data. For example, although the letter pair *sh* is usually pronounced as a single phoneme in English (e.g. *gash* [g æ ʃ]), this is not true universally (e.g. *gasholder* [g æ s h o l d ə r]). Therefore, we implement a letter chunk prediction model to provide chunk boundaries given only graphemes.

In our system, a bigram letter chunking prediction automatically discovers double letters based on instance-based learning (Aha et al., 1991). Since the many-to-many alignments are drawn from 1-0, 1-1, 1-2, 2-0, and 2-1 relationships, each letter in a word can form a chunk with its neighbor or stand alone as a chunk itself. We treat the chunk prediction as a binary classification problem. We generate all the bigrams in a word and determine whether each bigram should be a chunk based on its context. Table 1 shows an example of how chunking prediction proceeds for the word *longs*. Letters $l_{i-2}, l_{i-1}, l_{i+1}$, and l_{i+2} are the context of the bigram l_i; $chunk = 1$ if the letter bigram l_i is a chunk. Otherwise, the chunk simply consists of an individual letter. In the example, the word is decomposed as *l|o|ng|s*, which can be aligned with its pronunciation [l | ɒ | ŋ | z]. If the model happens to predict consecutive overlapping chunks, only the first of the two is accepted.

l_{i-2}	l_{i-1}	l_i	l_{i+1}	l_{i+2}	$chunk$
_	_	lo	n	g	0
_	l	on	g	s	0
l	o	ng	s	_	1
o	n	gs	_	_	0

Table 1: An example of letter chunking prediction.

4 Phoneme prediction

Most of the previously proposed techniques for phoneme prediction require training data to be aligned in one-to-one alignments. Those models approach the phoneme prediction task as a classification problem: a phoneme is predicted for each letter independently without using other predictions from the same word. These local predictions assume independence of predictions, even though there are clearly interdependencies between predictions. Predicting each phoneme in a word without considering other assignments may not satisfy the main goal of finding a set of phonemes that work together to form a word.

A trigram phoneme prediction with constraint satisfaction inference (Van Den Bosch and Canisius, 2006) was proposed to improve on local predictions. From each letter unit, it predicts a trigram class that has the target phoneme in the middle surrounded by its neighboring phonemes. The phoneme sequence is generated in such a way that it satisfies the trigram, bigram and unigram constraints. The overlapping predictions improve letter-to-phoneme performance mainly by repairing imperfect one-to-one alignments.

However, the trigram class prediction tends to be more complex as it increases the number of target classes. For English, there are only 58 unigram phoneme classes but 13,005 tri-gram phoneme classes. The phoneme combinations in the tri-gram classes are potentially confusing to the prediction model because the model has more target classes in its search space while it has access to the same number of local features in the grapheme side.

We propose to apply a supervised HMM method embedded with local classification to find the most likely sequence of phonemes given a word. An HMM is a statistical model that combines the observation likelihood (probability of phonemes given letters) and transition likelihood (probability of current phoneme given previous phonemes) to predict each phoneme. Our approach differs from a basic Hidden Markov Model for letter-to-phoneme system (Taylor, 2005) that formulates grapheme sequences as observation states and phonemes as hidden states. The basic HMM system for L2P does not provide good performance on the task because it lacks context information on the grapheme side. In fact, a pronunciation depends more on graphemes than on the neighboring phonemes; therefore, the transition probability (language model) should affect the prediction decisions only when there is more than one possible phoneme that can be assigned to a letter.

Our approach is to use an instance-based learning technique as a local predictor to generate a set of phoneme candidates for each letter chunk, given its context in a word. The local predictor produces confidence values for Each candidate phoneme. We normalize the confidence values into values between 0 and 1, and treat them as the emission probabilities, while the transition probabilities are derived directly from the phoneme sequences in the training data.

The pronunciation is generated by considering both phoneme prediction values and transition probabilities. The optimal phoneme sequence is found with the Viterbi search algorithm. We limit the size of the context to $n = 3$ in order to avoid overfitting and minimize the complexity of the model. Since the candidate set is from the classifier, the search space is limited to a small number of candidate phonemes (1 to 5 phonemes in most cases).

The HMM postprocessing is independent of local predictions from the classifier. Instead, it selects the best phoneme sequence from a set of possible local predictions by taking advantage of the phoneme language model, which is trained on the phoneme sequences in the training data.

5 Evaluation

We evaluated our approaches on CMUDict, Brulex, and German, Dutch and English Celex corpora (Baayen et al., 1996). The corpora (except English Celex) are available as part of the Letter-to-Phoneme Conversion PRONALSYL Challenge[1].

[1]The PRONALSYL Challenge: `http://www.pascal-network.org/Challenges/PRONALSYL/`.

Language	Data set	Number of words
English	CMUDict	112,102
English	Celex	65,936
Dutch	Celex	116,252
German	Celex	49,421
French	Brulex	27,473

Table 2: Number of words in each data set.

For the English Celex data, we removed duplicate words as well as words shorter than four letters. Table 2 shows the number of words and the language of each corpus.

For all of our experiments, our local classifier for predicting phonemes is the instance-based learning IB1 algorithm (Aha et al., 1991) implemented in the TiMBL package (Daelemans et al., 2004). The HMM technique is applied as post processing to the instance-based learning to provide a sequence prediction. In addition to comparing one-to-one and many-to-many alignments, we also compare our method to the constraint satisfaction inference method as described in Section 4. The results are reported in word accuracy rate based on the 10-fold cross validation, with the mean and standard deviation values.

Table 3 shows word accuracy performance across a variety of methods. We show results comparing the one-to-one aligner described in Section 2.1 and the one-to-one aligner provided by the PRONALSYL challenge. The PRONALSYS one-to-one alignments are taken directly from the PRONALSYL challenge, whose method is based on an EM algorithm. For both alignments, we use instance-based learning as the prediction model.

Overall, our one-to-one alignments outperform the alignments provided by the data sets for all corpora. The main difference between the PRONALSYS one-to-one alignment and our one-to-one alignment is that our aligner does not allow a null letter on the grapheme side. Consider the word *abomination* [ə b ɒ m ɪ n e ʃ ə n]: the first six letters and phonemes are aligned the same way by both aligners (*abomin-* [ə b ɒ m ɪ n]). However, the two aligners produce radically different alignments for the last five letters. The alignment provided by the PRONALSYS one-to-one alignments is:

```
_  _  a  t  i  o  n
|  |  |  |  |  |  |
e  ʃ  ə  _  _  _  n
```

while our one-to-one alignment is:

```
a  t  i  o  n
|  |  |  |  |
e  _  ʃ  ə  n
```

Clearly, the latter alignment provides more information on how the graphemes map to the phonemes.

Table 3 also shows that impressive improvements for all evaluated corpora are achieved by using many-to-many alignments rather than one-to-one alignments (1-1 align vs. M-M align). The significant improvements, ranging from 2.7% to 7.6% in word accuracy, illustrate the importance of having more precise alignments. For example, we can now obtain the correct alignment for the second part of the word *abomination*:

```
a  ti  o  n
|  |   |  |
e  ʃ   ə  n
```

Instead of adding a null phoneme in the phoneme sequence, the many-to-many aligner maps the letter chunk *ti* to a single phoneme.

The HMM approach is based on the same hypothesis as the constraint satisfaction inference (CSInf) (Van Den Bosch and Canisius, 2006). The results in Table 3 (1-1+CSInf vs. 1-1+HMM) show that the HMM approach consistently improves performance over the baseline system (1-1 align), while the CSInf degrades performance on the Brulex data set. For the CSInf method, most errors are caused by trigram confusion in the prediction phase.

The results of our best system, which combines the HMM method with the many-to-many alignments (M-M+HMM), are better than the results reported in (Black et al., 1998) on both the CMUDict and German Celex data sets. This is true even though Black et al. (1998) use explicit lists of letter-phoneme mappings during the alignment process, while our approach is a fully automatic system that does not require any handcrafted list.

6 Conclusion and future work

We presented a novel technique of applying many-to-many alignments to the letter-to-phoneme conversion problem. The many-to-many alignments relax

Language	Data set	PRONALSYS	1-1 align	1-1+CsInf	1-1+HMM	M-M align	M-M+HMM
English	CMUDict	58.3 ± 0.49	60.3 ± 0.53	62.9 ± 0.45	62.1 ± 0.53	65.1 ± 0.60	65.6 ± 0.72
English	Celex	—	74.6 ± 0.80	77.8 ± 0.72	78.5 ± 0.76	82.2 ± 0.63	83.6 ± 0.63
Dutch	Celex	84.3 ± 0.34	86.6 ± 0.36	87.5 ± 0.32	87.6 ± 0.34	91.1 ± 0.27	91.4 ± 0.24
German	Celex	86.0 ± 0.40	86.6 ± 0.54	87.6 ± 0.47	87.6 ± 0.59	89.3 ± 0.53	89.8 ± 0.59
French	Brulex	86.3 ± 0.67	87.0 ± 0.38	86.5 ± 0.68	88.2 ± 0.39	90.6 ± 0.57	90.9 ± 0.45

Table 3: Word accuracies achieved on data sets based on the 10-fold cross validation. **PRONALSYS:** one-to-one alignments provided by the PRONALSYL challenge. **1-1 align:** our one-to-one alignment method described in Section 2.1. **CsInf:** Constraint satisfaction inference (Van Den Bosch and Canisius, 2006). **M-M align:** our many-to-many alignment method. **HMM:** our HMM embedded with a local prediction.

the constraint assumptions of the traditional one-to-one alignments. Letter chunking bigram prediction incorporates many-to-many alignments into the conventional phoneme prediction models. Finally, the HMM technique yields global phoneme predictions based on language models.

Impressive word accuracy improvements are achieved when the many-to-many alignments are applied over the baseline system. On several languages and data sets, using the many-to-many alignments, word accuracy improvements ranged from 2.7% to 7.6%, as compared to one-to-one alignments. The HMM cooperating with the local predictions shows slight improvements when it is applied to the many-to-many alignments. We illustrated that the HMM technique improves the word accuracy more consistently than the constraint-based approach. Moreover, the HMM can be easily incorporated into the many-to-many alignment approach.

We are investigating the possibility of integrating syllabification information into our system. It has been reported that syllabification can potentially improve pronunciation performance in English (Marchand and Damper, 2005). We plan to explore other sequence prediction approaches, such as discriminative training methods (Collins, 2004), and sequence tagging with Support Vector Machines (SVM-HMM) (Altun et al., 2003) to incorporate more features (context information) into the phoneme generation model. We are also interested in applying our approach to other related areas such as morphology and transliteration.

Acknowledgements

We would like to thank Susan Bartlett, Colin Cherry, and other members of the Natural Language Processing research group at University of Alberta for their helpful comments and suggestions. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden Markov Support Vector Machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*.

Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. The CELEX2 lexical database. LDC96L14.

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*, pages 77–80.

Michael Collins. 2004. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Langauge Processing (EMNLP)*.

Walter Daelemans and Antal Van Den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In *Progress in Speech Synthesis*, pages 77–89. Springer, New York.

Walter Daelemans, Jakub Zavrel, Ko Van Der Sloot, and Antal Van Den Bosch. 2004. TiMBL: Tilburg Memory Based Learner, version 5.1, reference guide. In *ILK Technical Report Series 04-02*.

Robert I. Damper, Yannick Marchand, John DS. Marsters, and Alexander I. Bazin. 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160, June.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. In *Journal of the Royal Statistical Society*, pages B:1–38.

Yannick Marchand and Robert I. Damper. 2000. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219, June.

Yannick Marchand and Robert I. Damper. 2005. Can syllabification improve pronunciation by analogy of English? In *Natural Language Engineering*, pages (1):1–25.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

David Sankoff and Joseph Kruskal, 1999. *Time Warps, String Edits, and Macromolecules*, chapter 2, pages 55–91. CSLI Publications.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. In *Complex Systems*, pages 1:145–168.

Paul Taylor. 2005. Hidden Markov Models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology 2005*.

Kristina Toutanova and Robert C. Moore. 2001. Pronunciation modeling for improved spelling correction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151, Morristown, NJ, USA. Association for Computational Linguistics.

Antal Van Den Bosch and Sander Canisius. 2006. Improved morpho-phonological sequence processing with constraint satisfaction inference. *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON '06*, pages 41–49, June.

Analysis of Morph-Based Speech Recognition and the Modeling of Out-of-Vocabulary Words Across Languages

Mathias Creutz[*], **Teemu Hirsimäki**[*], **Mikko Kurimo**[*], **Antti Puurula**[*], **Janne Pylkkönen**[*], **Vesa Siivola**[*], **Matti Varjokallio**[*], **Ebru Arısoy**[†], **Murat Saraçlar**[†], and **Andreas Stolcke**[‡]
[*] Helsinki University of Technology, `<firstname>.<lastname>@tkk.fi`,
[†] Boğaziçi University, `arisoyeb@boun.edu.tr, murat.saraclar@boun.edu.tr`,
[‡] SRI International / International Computer Science Institute, `stolcke@speech.sri.com`

Abstract

We analyze subword-based language models (LMs) in large-vocabulary continuous speech recognition across four "morphologically rich" languages: Finnish, Estonian, Turkish, and Egyptian Colloquial Arabic. By estimating n-gram LMs over sequences of $morphs$ instead of words, better vocabulary coverage and reduced data sparsity is obtained. Standard word LMs suffer from high out-of-vocabulary (OOV) rates, whereas the morph LMs can recognize previously unseen word forms by concatenating morphs. We show that the morph LMs generally outperform the word LMs and that they perform fairly well on OOVs without compromising the accuracy obtained for in-vocabulary words.

1 Introduction

As automatic speech recognition systems are being developed for an increasing number of languages, there is growing interest in language modeling approaches that are suitable for so-called "morphologically rich" languages. In these languages, the number of possible word forms is very large because of many productive morphological processes; words are formed through extensive use of, e.g., inflection, derivation and compounding (such as the English words 'rooms', 'roomy', 'bedroom', which all stem from the noun 'room').

For some languages, language modeling based on surface forms of words has proven successful, or at least satisfactory. The most studied language, English, is not characterized by a multitude of word forms. Thus, the recognition vocabulary can simply consist of a list of words observed in the training text, and n-gram language models (LMs) are estimated over word sequences. The applicability of the word-based approach to morphologically richer languages has been questioned. In highly compounding languages, such as the Germanic languages German, Dutch and Swedish, decomposition of compound words can be carried out to reduce the vocabulary size. Highly inflecting languages are found, e.g., among the Slavic, Romance, Turkic, and Semitic language families. LMs incorporating morphological knowledge about these languages can be applied. A further challenging category comprises languages that are both highly inflecting and compounding, such as the Finno-Ugric languages Finnish and Estonian.

Morphology modeling aims to reduce the out-of-vocabulary (OOV) rate as well as data sparsity, thereby producing more effective language models. However, obtaining considerable improvements in speech recognition accuracy seems hard, as is demonstrated by the fairly meager improvements (1–4 % relative) over standard word-based models accomplished by, e.g., Berton et al. (1996), Ordelman et al. (2003), Kirchhoff et al. (2006), Whittaker and Woodland (2000), Kwon and Park (2003), and Shafran and Hall (2006) for Dutch, Arabic, English, Korean, and Czech, or even the worse performance reported by Larson et al. (2000) for German and Byrne et al. (2001) for Czech. Nevertheless, clear improvements over a word baseline have been achieved for Serbo-Croatian (Geutner et al., 1998), Finnish, Estonian (Kurimo et al., 2006b) and Turkish (Kurimo et al., 2006a).

In this paper, subword language models in the recognition of speech of four languages are analyzed: Finnish, Estonian, Turkish, and the dialect of Arabic spoken in Egypt, Egyptian Colloquial Arabic (ECA). All these languages are considered "morphologically rich", but the benefits of using subword-based LMs differ across languages. We attempt to discover explanations for these differences. In particular, the focus is on the analysis of OOVs: A perceived strength of subword models, when contrasted with word models, is that subword models can generalize to previously unseen word forms by recognizing them as sequences of shorter familiar word fragments.

2 Morfessor

Morfessor is an unsupervised, data-driven, method for the segmentation of words into morpheme-like units. The general idea is to discover as compact a description of the input text corpus as possible. Substrings occurring frequently enough in several different word forms are proposed as *morphs*, and the words in the corpus are then represented as a concatenation of morphs, e.g., 'hand, hand+s, left+hand+ed, hand+ful'. Through maximum a posteriori optimization (MAP), an optimal balance is sought between the compactness of the inventory of morphs, i.e., the *morph lexicon*, versus the compactness of the representation of the corpus.

Among others, de Marcken (1996), Brent (1999), Goldsmith (2001), Creutz and Lagus (2002), and Creutz (2006) have shown that models based on the above approach produce segmentations that resemble linguistic morpheme segmentations, when formulated mathematically in a probabilistic framework or equivalently using the Minimum Description Length (MDL) principle (Rissanen, 1989). Similarly, Goldwater et al. (2006) use a hierarchical Dirichlet model in combination with morph bigram probabilities.

The Morfessor model has been developed over the years, and different model versions exist. The model used in the speech recognition experiments of the current paper is the original, so-called *Morfessor Baseline* algorithm, which is publicly available for download.[1]. The mathematics of the Morfessor Baseline model is briefly outlined in the following; consult Creutz (2006) for details.

[1] `http://www.cis.hut.fi/projects/morpho/`

2.1 MAP Optimization Criterion

In slightly simplified form, the optimization criterion utilized in the model corresponds to the maximization of the following posterior probability:

$$\begin{aligned} &P(\text{lexicon} \mid \text{corpus}) \propto \\ &P(\text{lexicon}) \cdot P(\text{corpus} \mid \text{lexicon}) = \\ &\prod_{\text{letters } \alpha} P(\alpha) \cdot \prod_{\text{morphs } \mu} P(\mu). \end{aligned} \quad (1)$$

The lexicon consists of all distinct morphs spelled out; this forms a long string of letters α, in which each morph is separated from the next morph using a morph boundary character. The probability of the lexicon is the product of the probability of each letter in this string. Analogously, the corpus is represented as a sequence of morphs, which corresponds to a particular segmentation of the words in the corpus. The probability of this segmentation equals the product of the probability of each morph token μ. Letter and morph probabilities are maximum likelihood estimates (empirical Bayes).

2.2 From Morphs to n-Grams

As a result of the probabilistic (or MDL) approach, the morph inventory discovered by the Morfessor Baseline algorithm is larger the more training data there is. In some speech recognition experiments, however, it has been desirable to restrict the size of the morph inventory. This has been achieved by setting a frequency threshold on the words on which Morfessor is trained, such that the rarest words will not affect the learning process. Nonetheless, the rarest words can be split into morphs in accordance with the model learned, by using the Viterbi algorithm to select the most likely segmentation. The process is depicted in Figure 1.

2.3 Grapheme-to-Phoneme Mapping

The mapping between graphemes (letters) and phonemes is straightforward in the languages studied in the current paper. More or less, there is a one-to-one correspondence between letters and phonemes. That is, the spelling of a word indicates the pronunciation of the word, and when splitting the word into parts, the pronunciation of the parts in isolation does not differ much from the pronunciation of the parts in context. However, a few exceptions

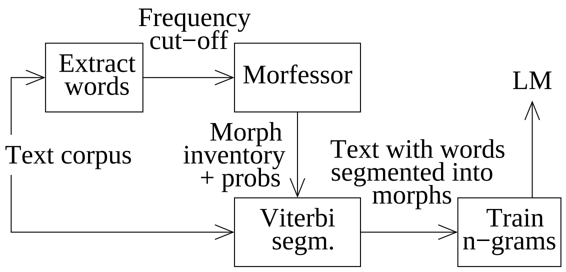


Figure 1: How to train a segmentation model using the Morfessor Baseline algorithm, and how to further train an n-gram model based on morphs.

have been treated more rigorously in the Arabic experiments: e.g., in some contexts the same (spelled) morph can have multiple possible pronunciations.

3 Experiments and Analysis

The goal of the conducted experiments is to compare n-gram language models based on morphs to standard word n-gram models in automatic speech recognition across languages.

3.1 Data Sets and Recognition Systems

The results from eight different tests have been analyzed. Some central properties of the test configurations are shown in Table 1. The Finnish, Estonian, and Turkish test configurations are slight variations of experiments reported earlier in Hirsimäki et al. (2006) (Fin1: 'News task', Fin2: 'Book task'), Kurimo et al. (2006a) (Fin3, Tur1), and Kurimo et al. (2006b) (Fin4, Est, Tur2).

Three different recognition platforms have been used, all of which are state-of-the-art large vocabulary continuous speech recognition (LVCSR) systems. The Finnish and Estonian experiments have been run on the HUT speech recognition system developed at Helsinki University of Technology.

The Turkish tests were performed using the AT&T decoder (Mohri and Riley, 2002); the acoustic features were produced using the HTK front end (Young et al., 2002). The experiments on Egyptian Colloquial Arabic (ECA) were carried out using the SRI Decipher^{TM} speech recognition system.

3.1.1 Speech Data and Acoustic Models

The type and amount of speech data vary from one language to another. The Finnish data consists of news broadcasts read by one single female speaker (Fin1), as well as an audio book read by another female speaker (Fin2, Fin3, Fin4). The Finnish acoustic models are speaker dependent (SD). Monophones (mon) were used in the earlier experiments (Fin1, Fin2), but these were later replaced by cross-context triphones (tri).

The Estonian speech data has been collected from a large number of speakers and consists of sentences from newspapers as well as names and digits read aloud. The acoustic models are speaker-independent triphones (SI tri) adapted online using Cepstral Mean Subtraction and Constrained Maximum Likelihood Linear Regression. Also the Turkish acoustic training data contains speech from hundreds of speakers. The test set is composed of newspaper text read by one female speaker. Speaker-independent triphones are used as acoustic models.

The Finnish, Estonian, and Turkish data sets contain planned speech, i.e., written text read aloud. By contrast, the Arabic data consists of transcribed spontaneous telephone conversations,[2] which are characterized by disfluencies and by the presence of "non-speech", such as laugh and cough sounds. There are multiple speakers in the Arabic data, and online speaker adaptation has been performed.

3.1.2 Text Data and Language Models

The n-gram language models are trained using the SRILM toolkit (Stolcke, 2002) (Fin1, Fin2, Tur1, Tur2, ECA) or similar software developed at HUT (Siivola and Pellom, 2005) (Fin3, Fin4, Est). All models utilize the Modified Interpolated Kneser-Ney smoothing technique (Chen and Goodman, 1999). The Arabic LM is trained on the same corpus that is used for acoustic training. This data set is regrettably small (160 000 words), but it matches the test set well in style, as it consists of transcribed spontaneous speech. The LM training corpora used for the other languages contain fairly large amounts of mainly news and book texts and conceivably match the style of the test data well.

In the morph-based models, words are split into morphs using Morfessor, and statistics are collected for morph n-grams. As the desired output of the

[2]LDC CallHome corpus of Egyptian Colloquial Arabic: `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC97S45`

Table 1: Test configurations

	Fin1	**Fin2**	**Fin3**	**Fin4**	**Est**	**Tur1**	**Tur2**	**ECA**
Recognizer	HUT	HUT	HUT	HUT	HUT	AT&T	AT&T	SRI
Speech data								
Type of speech	read	read	read	read	read	read	read	spont.
Training set [kwords]	20	49	49	49	790	230	110	160
Speakers in training set	1	1	1	1	1300	550	250	310
Test set [kwords]	4.3	1.9	1.9	1.9	3.7	7.0	7.0	16
Speakers in test set	1	1	1	1	50	1	1	57
Text data								
LM training set [Mwords]	36	36	32	150	53	17	27	0.16
Models								
Acoustic models	SD mon	SD mon	SD tri	SD tri	SI tri	SI tri	SI tri	SI tri
Morph lexicon [kmorphs]	66	66	120	25	37	52	34	6.1
Word lexicon [kwords]	410	410	410	–	60	120	50	18
Out-of-vocabulary words								
OOV LM training set [%]	5.0	5.0	5.9	–	14	5.3	9.6	0.61
OOV test set [%]	5.0	7.2	7.3	–	19	5.5	12	9.9
New words in test set [%]	2.7	3.0	3.1	1.5	3.4	1.6	1.5	9.8

speech recognizer is a sequence of words rather than morphs, the LM explicitly models word breaks as special symbols occurring in the morph sequence.

For comparison, word n-gram models have been tested. The vocabulary cannot typically include every word form occurring in the training set (because of the large number of different words), so the most frequent words are given priority; the actual lexicon sizes used in each experiment are shown in Table 1. Any word not contained in the lexicon is replaced by a special out-of-vocabulary symbol.

As words and morphs are units of different length, their optimal performance may occur at different orders of the n-gram. The best order of the n-gram has been optimized on development test sets in the following cases: Fin1, Fin2, Tur1, ECA (4-grams for both morphs and words) and Tur2 (5-grams for morphs, 3-grams for words). The models have additionally been pruned using entropy-based pruning (Tur1, Tur2, ECA) (Stolcke, 1998). In the other experiments (Fin3, Fin4, Est), no fixed maximum value of n was selected. n-Gram growing was performed (Siivola and Pellom, 2005), such that those n-grams that maximize the training set likelihood are gradually added to the model. The unrestricted growth of the model is counterbalanced by an MDL-type complexity term. The highest order of n-grams accepted was 7 for Finnish and 8 for Estonian.

Note that the optimization procedure is neutral with respect to morphs vs. words. Roughly the same number of parameters are allowed in the resulting LMs, but typically the morph n-gram LMs are smaller than the corresponding word n-gram LMs.

3.1.3 Out-of-Vocabulary Words

Table 1 further shows statistics on out-of-vocabulary rates in the data sets. This is relevant for the assessment of the word models, as the OOV rates define the limits of these models.

The OOV rate for the LM training set corresponds to the proportion of words replaced by the OOV symbol in the LM training data, i.e., words that were not included in the recognition vocabulary. The high OOV rates for Estonian (14 %) and Tur2 (9.6 %) indicate that the word lexicons have poor coverage of these sets. By contrast, the ECA word lexicon covers virtually the entire training set vocabulary.

Correspondingly, the test set OOV rate is the proportion of words that occur in the data sets used for running the speech recognition tests, but that are missing from the recognition lexicons. This value is thus the *minimum error* that can be obtained by the word models, or put differently, the recognizer is guaranteed to get at least this proportion of words wrong. Again, the values are very high for Estonian (19 %) and Tur2 (12 %), but also for Arabic (9.9 %) because of the insufficient amount of training data.

Finally, the figures labeled “new words in test set” denote the proportion of words in the test set that do not occur in the LM training set. Thus, these values indicate the minimum error achievable by *any* word model trained on the training sets available.

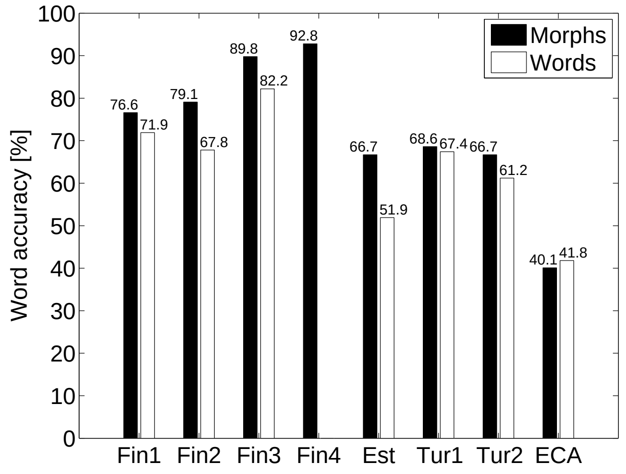


Figure 2: Word accuracies for the different speech recognition test configurations.

3.2 Results and Analysis

The morph-based and word-based results of the conducted speech recognition experiments are shown in Figure 2 (for Fin4, no comparable word experiment has been carried out). The evaluation measure used is *word accuracy* (WAC): the number of correctly recognized words minus the number of incorrectly inserted words divided by the number of words in the reference transcript. (Another frequently used measure is the *word error rate*, WER, which relates to word accuracy as WER = 100 % – WAC.)

Figure 2 shows that the morph models perform better than the word models, with the exception of the Arabic experiment (ECA), where the word model outperforms the morph model. The statistical significance of these differences is confirmed by one-tailed paired Wilcoxon signed-rank tests at the significance level of 0.05.

Overall, the best performance is observed for the Finnish data sets, which is explained by the speaker-dependent acoustic models and clean noise conditions. The Arabic setup suffers from the insufficient amount of LM training data.

3.2.1 In-Vocabulary Words

For a further investigation of the outcome of the experiments, the test sets have been partitioned into regions based on the types of words they contain. The recognition output is aligned with the reference transcript, and the regions aligned with *in-vocabulary* (IV) reference words (words contained in the vocabulary of the word model) are put in one partition and the remaining words (OOVs) are put in another partition. Word accuracies are then computed separately for the two partitions. Inserted words, i.e., words that are not aligned with any word in the reference, are put in the IV partition, unless they are adjacent to an OOV region, in which case they are put in the OOV partition.

Figure 3a shows word accuracies for the in-vocabulary words. Without exception, the accuracy for the IVs is higher than that of the entire test set vocabulary. One could imagine that the word models would do better than the morph models on the IVs, since the word models are totally focused on these words, whereas the morph models reserve modeling capacity for a much larger set of words. The word accuracies in Fig. 3a also partly seem to support this view. However, Wilcoxon signed-rank tests (level 0.05) show that the superiority of the word model is statistically significant only for Arabic and for Fin3.

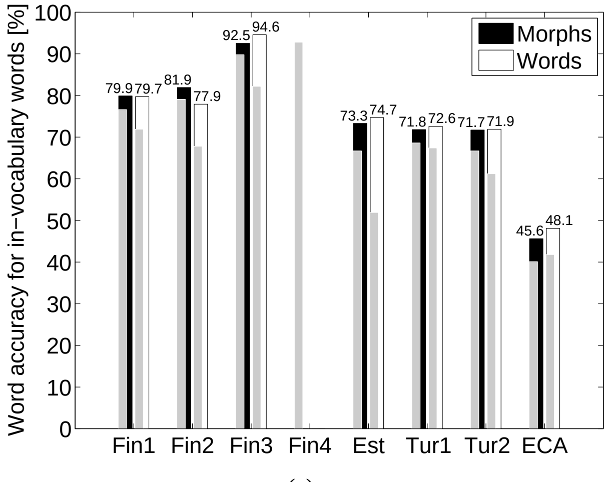
With few exceptions, it is thus possible to draw the conclusion that *morph models are capable of modeling a much larger set of words than word models without, however, compromising the performance on the limited vocabulary covered by the word models in a statistically significant way*.

3.2.2 Out-of-Vocabulary Words

Since the word model and morph model perform equally well on the subset of words that are included in the lexicon of the word model, the overall superiority of the morph model needs to come from its successful coping with out-of-vocabulary words.

In Figure 3b, word accuracies have been plotted for the out-of-vocabulary words contained in the test set. It is clear that the recognition accuracy for the OOVs is much lower than the overall accuracy. Also, negative accuracy values are observed. This happens when the number of insertions exceeds the number of correctly recognized units.

In Figure 3b, if speaker-dependent and speaker-independent setups are considered separately (and Arabic is left out), there is a tendency for the morph models to recognize the OOVs more accurately, the higher the OOV rate is. One could say that a morph model has a double advantage over a corresponding word model: the larger the proportion of OOVs



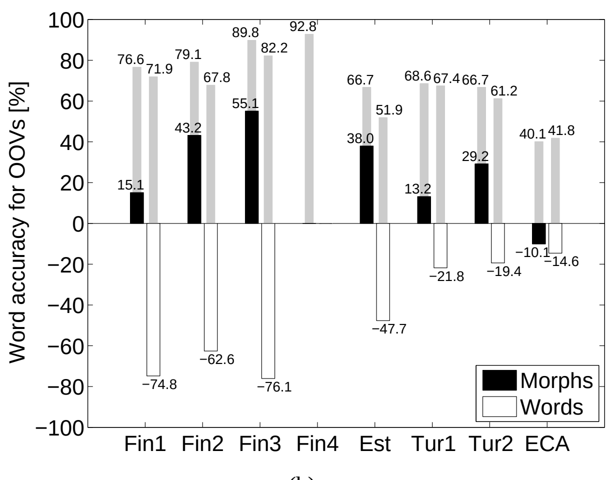


Figure 3: Word accuracies computed separately for those words in the test sets that are (a) included in and (b) excluded from the vocabularies of the word vocabulary; cf. figures listed on the row "OOV test set" in Table 1. Together these two partitions make up the entire test set vocabulary. For comparison, the results for the entire sets are shown using gray-shaded bars (also displayed in Figure 2).

in the word model is, the larger the proportion of words that the morph model can recognize but the word model cannot, a priori. In addition, the larger the proportion of OOVs, the more frequent and more "easily modelable" words are left out of the word model, and the more successfully these words are indeed learned by the morph model.

3.2.3 New Words in the Test Set

All words present in the training data (some of which are OOVs in the word models) "leave some trace" in the morph models, in the n-gram statistics that are collected for morph sequences. How, then, about new words that occur only in the test set, but not in the training set? In order to recognize such words correctly, the model must combine morphs in ways it has not observed before.

Figure 4 demonstrates that the new unseen words are very challenging. Now, also the morph models mostly obtain negative word accuracies, which means that the number of insertions adjacent to new words exceeds the number of correctly recognized new words. The best results are obtained in clean acoustic conditions (Fin2, Fin3, Fin4) with only few foreign names, which are difficult to get right using typical Finnish phoneme-to-grapheme mappings (as the negative accuracy of Fin1 suggests).

3.3 Vocabulary Growth and Arabic

Figure 5 shows the development of the size of the vocabulary (unique word forms) for growing amounts of text in different corpora. The corpora used for Finnish, Estonian, and Turkish (planned speech/text), as well as Arabic (spontaneous speech) are the LM training sets used in the experiments. Additional sources have been provided for Arabic and English: Arabic text (planned) from the FBIS corpus of Modern Standard Arabic (a collection of transcribed radio newscasts from various radio stations in the Arabic speaking world), as well as text from the New York Times magazine (English planned) and spontaneous transcribed English telephone conversations from the Fisher corpus.

The figure illustrates two points: (1) The faster the vocabulary growth is, the larger the potential advantage of morph models is in comparison to standard word models, because of OOV and data sparsity problems. The obtained speech recognition results seem to support this hypothesis; the applied morph LMs are clearly beneficial for Finnish and Estonian, mostly beneficial for Turkish, and slightly detrimental for ECA. (2) A more slowly growing vocabulary is used in spontaneous speech than in planned speech (or written text). Moreover, the Arabic 'spontaneous' curve is located fairly close

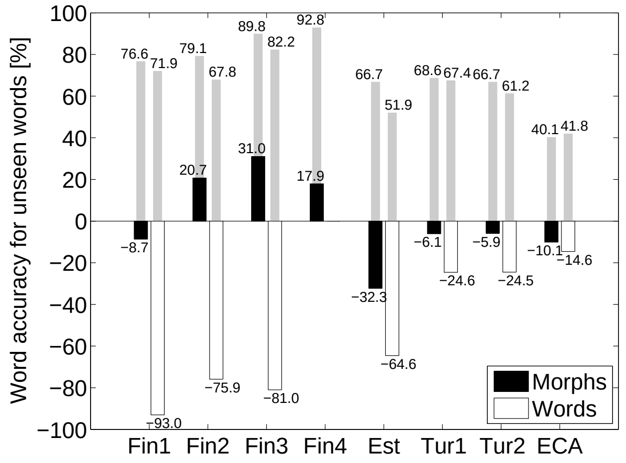


Figure 4: Word accuracies computed for the words in the test sets that do not occur at all in the training sets; cf. figures listed on the row "new words in test set" in Table 1. For comparison, the gray-shaded bars show the corresponding results for the entire test sets (also displayed in Figure 2).

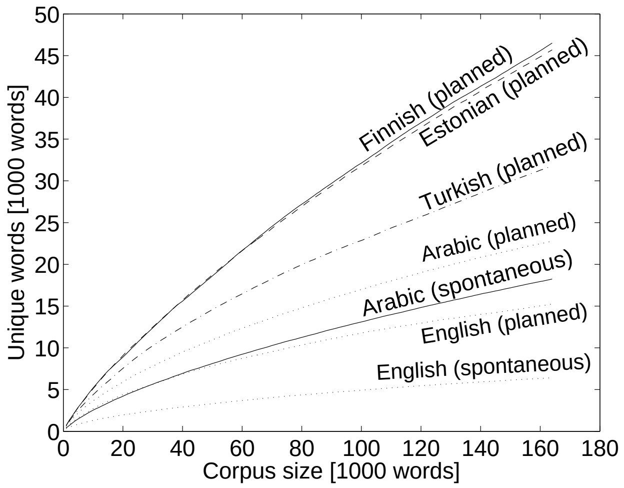


Figure 5: Vocabulary growth curves for the different corpora of spontaneous and planned speech (or written text). For growing amounts of text (word tokens) the number of unique different word forms (word types) occurring in the corpus are plotted.

to the English 'planned' curve and much below the Finnish, Estonian, and Turkish curves. Thus, even though Arabic is considered a "morphologically rich" language, this is not manifested through a considerable vocabulary growth (and high OOV rate) in the Egyptian Colloquial Arabic data used in the current speech recognition experiments. Consequently, it may not be that surprising that the morph model did not work particularly well for Arabic.

Arabic words consist of a stem surrounded by prefixes and suffixes, which are fairly successfully segmented out by Morfessor. However, Arabic also has *templatic* morphology, i.e., the stem is formed through the insertion of a vowel pattern into a "consonantal skeleton".

Additional experiments have been performed using the ECA data and Factored Language Models (FLMs) (Kirchhoff et al., 2006). The FLM is a powerful model that makes use of several sources of information, in particular a morphological lexicon of ECA. The FLM incorporates mechanisms for handling templatic morphology, but despite its sophistication, it barely outperforms the standard word model: The word accuracy of the FLM is 42.3 % and that of the word model is 41.8 %. The speech recognition implementation of both the FLM and the word model is based on *whole words* (although subword units are used for assigning probabilities to word forms in the FLM). This contrasts these models with the morph model, which splits words into subword units also in the speech recognition implementation. It seems that the splitting is a source of errors in this experimental setup with very little data available.

4 Discussion

Alternative morph-based and word-based approaches exist. We have tried some, but none of them has outperformed the described morph models for Finnish, Estonian, and Turkish, or the word and FLM models for Egyptian Arabic (in a statistically significant way). The tested models comprise more linguistically accurate morph segmentations obtained using later Morfessor versions (Categories-ML and Categories-MAP) (Creutz, 2006), as well as analyses obtained from morphological parsers.

Hybrids, i.e., word models augmented with phonemes or other subword units have been proposed (Bazzi and Glass, 2000; Galescu, 2003; Bisani and Ney, 2005). In our experiments, such models have outperformed the standard word models, but not the morph models.

Simply growing the word vocabulary to cover the entire vocabulary of large training corpora could be one (fairly "brute-force") approach, but this is hardly feasible for languages such as Finnish. The entire Finnish LM training data of 150 million words (used in Fin4) contains more than 4 million unique word forms, a value ten times the size of the rather large word lexicon currently used. And even if a 4-million-word lexicon were to be used, the OOV rate of the test set would still be relatively high: 1.5 %.

Judging by the Arabic experiments, there seems to be some potential in Factored Language Models. The FLMs might work well also for the other languages, and in fact, to do justice to the more advanced morph models from later versions of Morfessor, FLMs or some other refined techniques may be necessary as a complement to the currently used standard n-grams.

Acknowledgments

We are most grateful to Katrin Kirchhoff and Dimitra Vergyri for their valuable help on issues related to Arabic, and to the EU AMI training program for funding part of this work. The work was also partly funded by DARPA under contract No. HR0011-06-C-0023 (approved for public release, distribution is unlimited). The views herein are those of the authors and do not necessarily reflect the views of the funding agencies.

References

I. Bazzi and J. R. Glass. 2000. Modeling out-of-vocabulary words for robust speech recognition. In *Proc. ICSLP*, Beijing, China.

A. Berton, P. Fetter, and P. Regel-Brietzmann. 1996. Compound words in large-vocabulary German speech recognition systems. In *Proc. ICSLP*, pp. 1165–1168, Philadelphia, PA, USA.

M. Bisani and H. Ney. 2005. Open vocabulary speech recognition with flat hybrid models. In *Proc. Interspeech*, Lisbon, Portugal.

M. R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

W. Byrne, J. Hajič, P. Ircing, F. Jelinek, S. Khudanpur, P. Krbec, and J. Psutka. 2001. On large vocabulary continuous speech recognition of highly inflectional language — Czech. In *Proc. Eurospeech*, pp. 487–489, Aalborg, Denmark.

S. F. Chen and J. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Proc. ACL SIGPHON*, pp. 21–30, Philadelphia, PA, USA.

M. Creutz. 2006. *Induction of the Morphology of Natural Language: Unsupervised Morpheme Segmentation with Application to Automatic Speech Recognition*. Ph.D. thesis, Helsinki University of Technology. `http://lib.tkk.fi/Diss/2006/isbn9512282119/`.

C. G. de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, MIT.

L. Galescu. 2003. Recognition of out-of-vocabulary words with sub-lexical language models. In *Proc. Eurospeech*, pp. 249–252, Geneva, Switzerland.

P. Geutner, M. Finke, and P. Scheytt. 1998. Adaptive vocabularies for transcribing multilingual broadcast news. In *Proc. ICASSP*, pp. 925–928, Seattle, WA, USA.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

S. Goldwater, T. L. Griffiths, and M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proc. Coling/ACL*, pp. 673–680, Sydney, Australia.

T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*, 20(4):515–541.

K. Kirchhoff, D. Vergyri, J. Bilmes, K. Duh, and A. Stolcke. 2006. Morphology-based language modeling for Arabic speech recognition. *Computer Speech and Language*, 20(4):589–608.

M. Kurimo, M. Creutz, M. Varjokallio, E. Arısoy, and M. Saraçlar. 2006a. Unsupervised segmentation of words into morphemes – Morpho Challenge 2005, Application to automatic speech recognition. In *Proc. Interspeech*, Pittsburgh, PA, USA.

M. Kurimo, A. Puurula, E. Arısoy, V. Siivola, T. Hirsimäki, J. Pylkkönen, T. Alumäe, and M. Saraçlar. 2006b. Unlimited vocabulary speech recognition for agglutinative languages. In *Proc. NAACL-HLT*, New York, USA.

O.-W. Kwon and J. Park. 2003. Korean large vocabulary continuous speech recognition with morpheme-based recognition units. *Speech Communication*, 39(3–4):287–300.

M. Larson, D. Willett, J. Koehler, and G. Rigoll. 2000. Compound splitting and lexical unit recombination for improved performance of a speech recognition system for German parliamentary speeches. In *Proc. ICSLP*.

M. Mohri and M. D. Riley. 2002. DCD library, Speech recognition decoder library. AT&T Labs Research. `http://www.research.att.com/sw/tools/dcd/`.

R. Ordelman, A. van Hessen, and F. de Jong. 2003. Compound decomposition in Dutch large vocabulary speech recognition. In *Proc. Eurospeech*, pp. 225–228, Geneva, Switzerland.

J. Rissanen. 1989. Stochastic complexity in statistical inquiry. *World Scientific Series in Computer Science*, 15:79–93.

I. Shafran and K. Hall. 2006. Corrective models for speech recognition of inflected languages. In *Proc. EMNLP*, Sydney, Australia.

V. Siivola and B. Pellom. 2005. Growing an n-gram model. In *Proc. Interspeech*, pp. 1309–1312, Lisbon, Portugal.

A. Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA BNTU Workshop*, pp. 270–274, Lansdowne, VA, USA.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. ICSLP*, pp. 901–904. `http://www.speech.sri.com/projects/srilm/`.

E. W. D. Whittaker and P. C. Woodland. 2000. Particle-based language modelling. In *Proc. ICSLP*, pp. 170–173, Beijing, China.

S. Young, D. Ollason, V. Valtchev, and P. Woodland. 2002. *The HTK book (for version 3.2 of HTK)*. University of Cambridge.

Tree Revision Learning for Dependency Parsing

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
attardi@di.unipi.it

Massimiliano Ciaramita
Yahoo! Research Barcelona
Barcelona, Spain
massi@yahoo-inc.com

Abstract

We present a revision learning model for improving the accuracy of a dependency parser. The revision stage corrects the output of the base parser by means of revision rules learned from the mistakes of the base parser itself. Revision learning is performed with a discriminative classifier. The revision stage has linear complexity and preserves the efficiency of the base parser. We present empirical evaluations on the treebanks of two languages, which show effectiveness in relative error reduction and state of the art accuracy.

1 Introduction

A dependency parse tree encodes useful semantic information for several language processing tasks. Dependency parsing is a simpler task than constituent parsing, since dependency trees do not have extra non-terminal nodes and there is no need for a grammar to generate them. Approaches to dependency parsing either generate such trees by considering all possible spanning trees (McDonald et al., 2005), or build a single tree on the fly by means of shift-reduce parsing actions (Yamada & Matsumoto, 2003). In particular, Nivre and Scholz (2004) and Attardi (2006) have developed deterministic dependency parsers with linear complexity, suitable for processing large amounts of text, as required, for example, in information retrieval applications.

We investigate a novel revision approach to dependency parsing related to re-ranking and transformation-based methods (Brill, 1993; Brill, 1995; Collins, 2000; Charniak & Johnson, 2005; Collins & Koo, 2006). Similarly to re-ranking, the second stage attempts to improve the output of a base parser. Instead of re-ranking n-best candidate parses, our method works by revising a single parse tree, either the $first$-best or the one constructed by a deterministic shift-reduce parser, as in transformation-based learning. Parse trees are revised by applying rules which replace incorrect with correct dependencies. These rules are learned by comparing correct parse trees with incorrect trees produced by the base parser on a training corpus. We use the same training corpus on which the base parser was trained, but this need not be the case. Hence, we define a new learning task whose output space is a set of revision rules and whose input is a set of features extracted at each node in the parse trees produced by the parser on the training corpus. A statistical classifier is trained to solve this task.

The approach is more suitable for dependency parsing since trees do not have non-terminal nodes, therefore revisions do not require adding/removing nodes. However, the method applies to any parser since it only analyzes output trees. An intuitive motivation for this method is the observation that a dependency parser correctly identifies most of the dependencies in a tree, and only local corrections might be necessary to produce a correct tree. Performing several parses in order to generate multiple trees would often just repeat the same steps. This could be avoided by focusing on the points where attachments are incorrect. In the experiments reported below, on average, the revision stage performs 4.28

corrections per sentence, or one every 6.25 tokens.

In our implementation we adopt a shift-reduce parser which minimizes computational costs. The resulting two-stage parser has complexity $O(n)$, linear in the length of the sentence. We evaluated our model on the treebanks of English and Swedish. The experimental results show a relative error reduction of, respectively, 16% and 11% with respect to the base parser, achieving state of accuracy on Swedish.

2 Dependency parsing

Detection of dependency relations can be useful in tasks such as information extraction (Culotta & Sorensen, 2004), lexical acquisition (Snow et al., 2005), ontology learning (Ciaramita et al., 2005), and machine translation (Ding & Palmer, 2005). A dependency parser is trained on a corpus annotated with lexical dependencies, which are easier to produce by annotators without deep linguistic knowledge and are becoming available in many languages (Buchholz & Marsi, 2006). Recent developments in dependency parsing show that deterministic parsers can achieve good accuracy (Nivre & Scholz, 2004), and high performance, in the range of hundreds of sentences per second (Attardi, 2006).

A dependency parser takes as input a sentence s and returns a dependency graph G. Let $D = \{d_1, d_2, ..., d_m\}$ be the set of permissible dependency types. A *dependency graph* for a sentence $s = \langle s_1, s_2, ..., s_n \rangle$ is a labeled directed graph $G = (s, A)$, such that:

(a) s is the set of nodes, corresponding to the tokens in the input string;
(b) A is a set of labeled arcs (w_i, d, w_j), $w_{i,j} \in s$, $d \in D$; w_j is called the *head*, w_i the *modifier* and d the *dependency label*;
(c) $\forall w_i \in s$ there is at most one arc $a \in A$, such that $a = (w_i, d, w_j)$;
(d) there are no cycles;

In statistical parsing a generator (e.g. a PCFG) is used to produce a number of candidate trees (Collins, 2000) with associated scores. This approach has been used also for dependency parsing, generating spanning trees as candidates and computing the maximum spanning tree using discriminative learning algorithms (McDonald et al., 2005).

$Shift$	$\frac{\langle S,n\|I,T,A\rangle}{\langle n\|S,I,T,A\rangle}$	(1)
$Right$	$\frac{\langle s\|S,n\|I,T,A\rangle}{\langle S,n\|I,T,A\cup\{(s,r,n)\}\rangle}$	(2)
$Left$	$\frac{\langle s\|S,n\|I,T,A\rangle}{\langle S,s\|I,T,A\cup\{(n,r,s)\}\rangle}$	(3)
$Right_2$	$\frac{\langle s_1\|s_2\|S,n\|I,T,A\rangle}{\langle s_1\|S,n\|I,T,A\cup\{(s_2,r,n)\}\rangle}$	(4)
$Left_2$	$\frac{\langle s_1\|s_2\|S,n\|I,T,A\rangle}{\langle s_2\|S,s_1\|I,T,A\cup\{(n,r,s_2)\}\rangle}$	(5)
$Right_3$	$\frac{\langle s_1\|s_2\|s_3\|S,n\|I,T,A\rangle}{\langle s_1\|s_2\|S,n\|I,T,A\cup\{(s_3,r,n)\}\rangle}$	(6)
$Left_3$	$\frac{\langle s_1\|s_2\|s_3\|S,n\|I,T,A\rangle}{\langle s_2\|s_3\|S,s_1\|I,T,A\cup\{(n,r,s_3)\}\rangle}$	(7)
$Extract$	$\frac{\langle s_1\|s_2\|S,n\|I,T,A\rangle}{\langle n\|s_1\|S,I,s_2\|T,A\rangle}$	(8)
$Insert$	$\frac{\langle S,I,s_1\|T,A\rangle}{\langle s_1\|S,I,T,A\rangle}$	(9)

Table 1. The set of parsing rules of the base parser.

Yamada and Matsumoto (2003) have proposed an alternative approach, based on deterministic bottom-up parsing. Instead of learning directly which tree to assign to a sentence, the parser learns which *Shift/Reduce* actions to use for building the tree. Parsing is cast as a classification problem: at each step the parser applies a classifier to the features representing its current state to predict the next action to perform. Nivre and Scholz (2004) proposed a variant of the model of Yamada and Matsumoto that reduces the complexity from the worst case quadratic to linear. Attardi (2006) proposed a variant of the rules that allows deterministic single-pass parsing and as well as handling non-projective relations. Several approaches to dependency parsing on multiple languages have been evaluated in the CoNLL-X Shared Task (Buchholz & Marsi, 2006).

3 A shift-reduce dependency parser

As a base parser we use DeSR, a shift-reduce parser described in (Attardi, 2006). The parser constructs dependency trees by scanning input sentences in a single left-to-right pass and performing Shift/Reduce parsing actions. The parsing algorithm is fully deterministic and has linear complexity. Its behavior can be described as repeatedly selecting and applying some parsing rules to transform its state.

The state of the parser is represented by a quadruple $\langle S, I, T, A\rangle$: S is the stack, I is the list of (remaining) input tokens, T is a stack of saved tokens and A is the arc relation for the dependency graph, consisting of a set of labeled arcs (w_i, r, w_j), $w_i, w_j \in W$ (the set of tokens), and $d \in D$ (the set of dependencies). Given an input sentence s, the parser is initialized to $\langle\emptyset, s, \emptyset, \emptyset\rangle$, and terminates when it reaches the configuration $\langle s, \emptyset, \emptyset, A\rangle$.

Table 1 lists all parsing rules. The $Shift$ rule advances on the input, while the various $Left$, $Right$ variants create links between the next input token and some previous token on the stack. $Extract/Insert$ generalize the previous rules by respectively moving one token to the stack T and reinserting the top of T into S. An essential difference with respect to the rules of Yamada and Matsumoto (2003) is that the $Right$ rules move back to the input the top of the stack, allowing some further processing on it, which would otherwise require a second pass. The extra $Left$ and $Right$ rules (4-7, Table 1), and the $ExtractInsert$ rules (8 and 9, Table 1), are new rules added for handling non-projective trees. The algorithm works as follows:

Algorithm 1: DeSR

```
input: $s = w_1, w_2, ..., w_n$
begin
    $S \leftarrow \langle\rangle$
    $I \leftarrow \langle w_1, w_2, ..., w_n\rangle$
    $T \leftarrow \langle\rangle$
    $A \leftarrow \langle\rangle$
    while $I \neq \langle\rangle$ do
        $\mathbf{x} \leftarrow getContext(S, I, T, A)$
        $y \leftarrow estimateAction(\mathbf{w}, \mathbf{x})$
        $performAction(y, S, I, T, A)$
end
```

The function $getContext()$ extracts a vector $\mathbf{x}$ of contextual features around the current token, i.e., from a subset of I and S. $estimateAction()$ predicts a parsing action y given a trained model $\mathbf{w}$ and $\mathbf{x}$. In the experiments presented below, we used as features the lemma, Part-of-Speech, and dependency type of the following items:

- 2 top items from S;
- 4 items from I;
- 2 leftmost and 2 rightmost children from the top of S and I.

Step	Description
r	Up to root node
u	Up one parent
$-n$	Left to the n-th token
$+n$	Right to the n-th token
[	Head of previous constituent
]	Head of following constituent
$>$	First token of previous constituent
$<$	First token of following constituent
$d--$	Down to the leftmost child
$d++$	Down to the rightmost child
$d-1$	Down to the first left child
$d+1$	Down to the first right child
dP	Down to token with POS P

Table 2. Description of the atomic movements allowed on the graph relatively to a token w.

4 Revising parse trees

The base parser is fairly accurate and even when there are mistakes most sentence chunks are correct. The full correct parse tree can often be recovered by performing just a small number of revisions on the base parse. We propose to learn these revisions and to apply them to the single best tree output by the base parser. Such an approach preserves the deterministic nature of the parser, since revising the tree requires a second sequential step over the whole sentence. The second step may also improve accuracy by incorporating additional evidence, gathered from the analysis of the tree which is not available during the first stage of parsing.

Our approach introduces a second learning task in which a model is trained to revise parse trees. Several questions needs to be addressed: which tree transformations to use in revising the parse tree, how to determine which transformation to apply, in which order, and which features to use for learning.

4.1 Basic graph movements

We define a revision as a combination of atomic moves on a graph; e.g., moving a link to the following or preceding token in the sentence, up or down the graph following the directed edges. Table 2 summarizes the set of atomic steps we used.

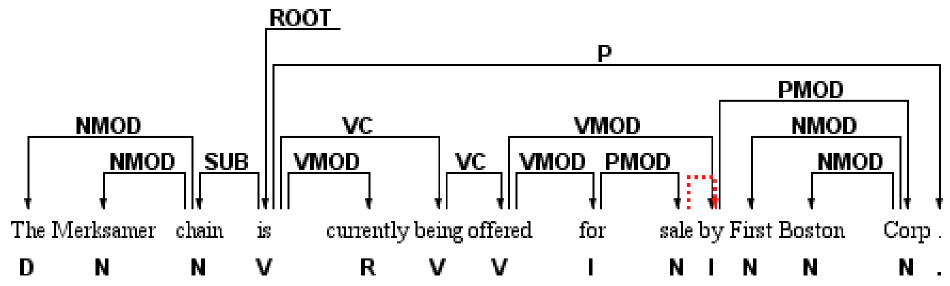

ROOT
P
PMOD
NMOD
VC
VMOD
NMOD
NMOD
SUB
VMOD
VC
VMOD
PMOD
NMOD
The Merksamer chain is currently being offered for sale by First Boston Corp .
D N N V R V V I N I N N N .

Figure 1. An incorrect dependency tree: the dashed arrow from "sale" to "by" should be replaced with the one from "offered" to "by".

4.2 Revision rules

A *revision rule* is a sequence of atomic steps on the graph which identifies the head of a modifier. As an example, Figure 1 depicts a tree in which the modifier "by" is incorrectly attached to the head "sale" (dashed arrow), rather than to the correct head "offered" (continuous arrow)[1]. There are several possible revision rules for this case: "uu", move up two nodes; -3, three tokens to the left, etc. To bound the complexity of feature extraction the maximum length of a sequence is bound to 4. A revision for a dependency relation is a link re-direction, which moves a single link in a tree to a different head. This is an elementary transformation which preserves the number of nodes in the tree.

A possible problem with these rules is that they are not tree-preserving, i.e. a tree may become a cyclic graph. For instance, rules that create a link to a descendant introduce cycles, unless the application of another rule will link one of the nodes in the path to the descendant to a node outside the cycle. To address these issues we apply the following heuristics in selecting the proper combination: rules that redirect to child nodes are chosen only when no other rule is applicable (upwards rule are safe), and shorter rules are preferred over longer ones. In our experiments we never observed the production of any cycles.

On Wall Street Journal Penn Treebank section 22 we found that the 20 most frequent rules are sufficient to correct 80% of the errors, see Table 3. This confirms that the atomic movements produce simple and effective revision rules.

[1]Arrows go from head to modifier as agreed among the participants to the CoNLL-X shared task.

COUNTS	RULE	TARGET LOCATION
983	uu	Up twice
685	-1	Token to the left
469	+1	Token to the right
265	[	Head of previous constituent
215	uuu	Up 3 times
197	+1u	Right, up
194	r	To root
174	-1u	Left, up
116	>u	Token after constituent, up
103	ud−−	Up down to leftmost child
90	V	To 1st child with POS verb
83	d+1	Down to first right child
82	uuuu	Up 4 times
74	<	Token before constituent
73	ud+1	Up down to 1st right child
71	uV	Up, down to 1st verb
61	ud-1	Up, down to last left child
56	ud+1d+1	Up, down to 1st right child twice
55	d+1d+1	Down to 1st right child twice
48	d−−	Down to leftmost child

Table 3. 20 most frequent revision rules in wsj22.

4.3 Tree revision problem

The tree revision problem can be formalized as follows. Let $G = (s, A)$ be a dependency tree for sentence $s = \langle w_1, w_2, ..., w_n \rangle$. A revision rule is a mapping $r : A \rightarrow A$ which, when applied to an arc $a = (w_i, d, w_j)$, returns an arc $a' = (w_i, d, w_s)$. A revised parse tree is defined as $r(G) = (s, A')$ such that $A' = \{r(a) : a \in A\}$.

This definition corresponds to applying the revisions to the original tree in a batch, as in (Brill, 1993). Alternatively, one could choose to apply the transformations incrementally, applying each one to the tree resulting from previous applications. We chose the first alternative, since the intermediate trees created during the transformation process may not be well-formed dependency graphs, and analyzing them in order to determine features for classification might incur problems. For instance, the graph might have abnormal properties that differ from those of any other graph produced by the parser. Moreover, there might not be enough cases of such graphs to form a sufficiently large training set.

5 Learning a revision model

We frame the problem of revising a tree as a supervised classification task. Given a training set $S = (x_i, y_i)_{i=1}^N$, such that $x_i \in \mathbb{R}^d$ and $y_i \in Y$, our goal is to learn a classifier, i.e., a function $F : X \rightarrow Y$. The output space represents the revision rules, in particular we denote with y_1 the identity revision rule. Features represents syntactic and morphological properties of the dependency being examined in its context on the graph.

5.1 Multiclass perceptron

The classifier used in revision is based on the perceptron algorithm (Rosemblatt, 1958), implemented as a multiclass classifier (Crammer & Singer, 2003). One introduces a weight vector $\alpha_i \in \mathbb{R}^d$ for each $y_i \in Y$, in which $\alpha_{i,j}$ represents the weight associated with feature j in class i, and learn α with the perceptron from the training data using a winner-take-all discriminant function:

$$F(x) = \arg\max_{y \in Y} \langle x, \alpha_y \rangle \qquad (10)$$

The only adjustable parameter in this model is the number of instances T to use for training. We chose T by means of validation on the development data, typically with a value around 10 times the size of the training data. For regularization purposes we adopt an average perceptron (Collins, 2002) which returns for each y, $\alpha_y = \frac{1}{T}\sum_{t=1}^{T} \alpha_y^t$, the average of all weight vectors α_y^t posited during training. The perceptron was chosen because outperformed other algorithms we experimented with (MaxEnt, MBL and SVM), particularly when including feature pairs, as discussed later.

5.2 Features

We used as features for the revision phase the same type of features used for training the parser (described in Section 3). This does not have to be the case in general. In fact, one might want to introduce features that are specific for this task. For example, global features of the full tree which might be not possible to represent or extract while parsing, as in statistical parse re-ranking (Collins & Koo, 2006).

The features used are lemma, Part-of-Speech, and dependency type of the following items: the current node, its parent, grandparent, great-grandparent, of the children thereof and, in addition, the previous and next tokens of the node. We also add as features all feature pairs that occurred more than 10 times, to reduce the size of the feature space. In alternative one could use a polynomial kernel. We preferred this option because, given the large size of the training data, a dual model is often impractical.

5.3 Revision model

Given a dependency graph $G = (s, A)$, for a sentence $s = \langle w_1, ..., w_n \rangle$, the revised tree is $R(G) = (s, A')$, where each dependency a'_i is equal to $F(a_i)$. In other words, the head in a_i has been changed, or not, according to the rule predicted by the classifier. In particular, we assume that revisions are independent of each other and perform a revision of a tree from left to right. As Table 3 suggests, there are many revision rules with low frequency. Rather than learning a huge classifier, for rules with little training data, we limit the number of classes to a value k. We experimented with values between 30 and 50, accounting for 98-99% of all rules, and eventually used 50, by experimenting with the development portion of the data. All rules that fall outside the threshold are collected in a single class y_0 of "unresolved" cases. If predicted, y_0, similarly to y_1, has no effect on the dependency.

Occasionally, in 59 sentences out of 2416 on section 23 of the Wall Street Journal Penn Treebank (Marcus et al., 1993), the shift-reduce parser fails to attach a node to a head, producing a disconnected graph. The disconnected node will appear as a root, having no head. The problem occurs most often on punctuations (66/84 on WSJ section 23), so it affects only marginally the accuracy scores (UAS, LAS) as computed in the CoNLL-X evaluation (Buchholz & Marsi, 2006). A final step of the revision deals with multiple roots, using a heuristic rule it selects one of the disconnected sub-trees as root, a verb, and attaches all sub-trees to it.

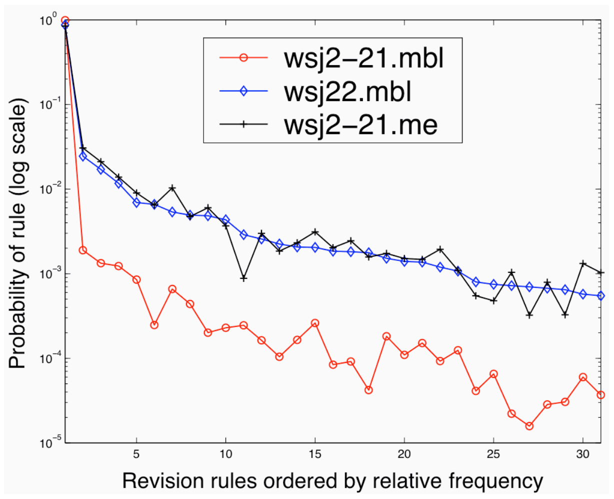


Figure 2. Frequency of the 30 most frequent rules obtained with different parsers on wsj22 and wsj2-21.

Parser	UAS	LAS
DeSR-ME	84.96	83.53
DeSR-MBL	88.41	86.85
Revision-MBL	89.11	86.39
Revision-ME	90.27	86.44
N&S	87.3	-
Y&M	90.3	-
MST-2	**91.5**	-

Table 4. Results on the Wall Street Journal Penn Treebank.

5.4 Algorithm complexity

The base dependency parser is deterministic and performs a single scan over the sentence. For each word it performs feature extraction and invokes the classifier to predict the parsing action. If prediction time is bound by a constant, as in linear classifiers, parsing has linear complexity. The revision pass is deterministic and performs similar feature extraction and prediction on each token. Hence, the complexity of the overall parser is $O(n)$. In comparison, the complexity of McDonald's parser (2006) is cubic, while the parser of Yamada and Matsumoto (2003) has a worst case quadratic complexity.

6 Experiments

6.1 Data and setup

We evaluated our method on English using the standard partitions of the Wall Street Journal Penn Treebank: sections 2-21 for training, section 22 for development, and section 23 for evaluation. The constituent trees were transformed into dependency trees by means of a script implementing rules proposed by Collins and Yamada[2]. In a second evaluation we used the Swedish Treebank (Nilsson et al., 2005) from CoNLL-X, approximately 11,000 sentences; for development purposes we performed cross-validation on the training data.

We trained two base parsers on the Penn Treebank: one with our own implementation of Maximum Entropy, one with the TiMBL library for Memory Based Learning (MBL, (Timbl, 2003)). We parsed sections 2 to 21 with each parser and produced two datasets for training the revision model: "wsj2-21.mbl" and "wsj2-21.me". Each dependency is represented as a feature vector (cf. Section 5.2), the prediction is a revision rule (cf. Section 4.2). For the smaller Swedish data we trained one base parser with MaxEnt and one with the SVM implementation in libSVM (Chang & Lin, 2001) using a polynomial kernel with degree 2.

[2]http://w3.msi.vxu.se/%7enivre/research/Penn2Malt.html

6.2 Results

On the Penn Treebank, the base parser trained with MBL (DeSR-MBL) achieves higher accuracy, 88.41 unlabeled accuracy score (UAS), than the same parser trained with MaxEnt (DeSR-ME), 84.96 UAS. The revision model trained on "wsj2-21.me" (Revision-ME) increases the accuracy of DeSR-ME to 88.01 UAS (+3%). The revision model trained on "wsj2-21.mbl" (DeSR-MBL) improves the accuracy of DeSR-MBL from 88.42 to 89.11 (+0.7%). The difference is mainly due to the fact that DeSR-MBL is quite accurate on the training data, almost 99%, hence "wsj2-21.mbl" contains less errors on which to train the revision parser. This is typical of the memory-based learning algorithm used in DeSR-MBL. Conversely, DeSR-ME achieves a score of of 85% on the training data, which is closer to the actual accuracy of the parser on unseen data. As an illustration, Figure 2 plots the distributions of revision rules in "wsj2-21.mbl" (DeSR-MBL), "wsj2-21.me" (DeSR-ME), and "wsj22.mbl" (DeSR-MBL) which represents the distribution of correct revision rules on the output of DeSR-MBL on the development set. The distributions of "wsj2-

Parser	UAS	LAS
DeSR-SVM	88.41	83.31
Revision-ME	**89.76**	83.13
Corston-Oliver& Aue	89.54	82.33
Nivre	89.50	84.58

Table 5. Results on the Swedish Treebank.

21.me" and "wsj22.mbl" are visibly similar, while "wsj2-21.mbl" is significantly more skewed towards not revising. Hence, the less accurate parser DeSR-ME might be more suitable for producing revision training data. Applying the revision model trained on "wsj2-21.me" (Revision-ME) to the output of DeSR-MBL the result is 90.27% UAS. A relative error reduction of 16.05% from the previous 88.41 UAS of DeSR-MBL. This finding suggests that it may be worth while experimenting with all possible revision-model/base-parser pairs as well as exploring alternative ways for generating data for the revision model; e.g., by cross-validation.

Table 4 summarizes the results on the Penn Treebank. Revision models are evaluated on the output of DeSR-MBL. The table also reports the scores obtained on the same data set by by the shift reduce parsers of Nivre and Scholz's (2004) and Yamada and Matsumoto (2003), and McDonald and Pereira's second-order maximum spanning tree parser (McDonald & Pereira, 2006). However the scores are not directly comparable, since in our experiments we used the settings of the CoNLL-X Shared Task, which provide correct POS tags to the parser.

On the Swedish Treebank collection we trained a revision model (Revision-ME) on the output of the MaxEnt base parser. We parsed the evaluation data with the SVM base parser (DeSR-SVM) which achieves 88.41 UAS. The revision model achieves 89.76 UAS, with a relative error reduction of 11.64%. Here we can compare directly with the best systems for this dataset in CoNLL-X. The best system (Corston-Oliver & Aue, 2006), a variant of the MST algorithm, obtained 89.54 UAS, while the second system (Nivre, 2006) obtained 89.50; cf. Table 5. Parsing the Swedish evaluation set (about 6,000 words) DeSR-SVM processes 1.7 words per second on a Xeon 2.8Ghz machine, DeSR-ME parses more than one thousand w/sec. In the revision step Revision-ME processes 61 w/sec.

7 Related work

Several authors have proposed to improve parsing via re-ranking (Collins, 2000; Charniak & Johnson, 2005; Collins & Koo, 2006). The base parser produces a list of n-best parse trees for a sentence. The re-ranker is trained on the output trees, using additional global features, with a discriminative model. These approaches achieve error reductions up to 13% (Collins & Koo, 2006). In transformation-based learning (Brill, 1993; Brill, 1995; Satta & Brill, 1995) the learning algorithm starts with a baseline assignment, e.g., the most frequent Part-of-Speech for a word, then repeatedly applies rewriting rules. Similarly to re-ranking our method aims at improving the accuracy of the base parser with an additional learner. However, as in transformation-based learning, it avoids generating multiple parses and applies revisions to arcs in the tree which it considers incorrect. This is consistent with the architecture of our base parser, which is deterministic and builds a single tree, rather than evaluating the best outcome of a generator.

With respect to transformation-based methods, our method does not attempt to build a tree but only to revise it. That is, it defines a different output space from the base parser's: the possible revisions on the graph. The revision model of Nakagawa et al. (2002) applies a second classifier for deciding whether the predictions of a base learner are accurate. However, the model only makes a binary decision, which is suitable for the simpler problem of POS tagging. The work of Hall and Novak (Hall & Novak, 2005) is the closest to ours. Hall and Novak develop a corrective model for constituency parsing in order to recover non-projective dependencies, which a standard constituent parser does not handle. The technique is applied to parsing Czech.

8 Conclusion

We presented a novel approach for improving the accuracy of a dependency parser by applying revision transformations to its parse trees. Experimental results prove that the approach is viable and promising. The proposed method achieves good accuracy and excellent performance using a deterministic shift-reduce base parser. As an issue for further investigation, we mention that in this framework, as in re-ranking, it is possible to exploit global features in the revision phase; e.g., semantic features such as those produced by named-entity detection systems.

Acknowledgments

We would like to thank Jordi Atserias and Brian Roark for useful discussions and comments.

References

G. Attardi. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proceedings of CoNNL-X 2006*.

S. Buchholz and E. Marsi. 2006. Introduction to CoNNL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNNL-X 2006*.

E. Brill. 1993. Automatic Grammar Induction and Parsing free Text: A Transformation-Based Approach. In *Proceedings of ACL 1993.*

E. Brill. 1995. *Transformation-Based Error-Driven Learning and Natural Language Processing.* Computational Linguistics 21(4): pp.543-565.

E. Charniak and M. Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of ACL 2005*.

C. Chang and C. Lin. 2001. LIBSVM: A Library for Support Vector Machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

M. Ciaramita, A. Gangemi, E. Ratsch, J. Sarić and I. Rojas. 2005. Unsupervised Learning of Semantic Relations between Concepts of a Molecular Biology Ontology. In *Proceedings of IJCAI 2005*.

M. Collins. 2000. Discriminative Reranking for Natural Language Parsing. In *Proceedings of ICML 2000.*

M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP 2002.*

M. Collins and T. Koo. 2006. *Discriminative Reranking for Natural Language Parsing.* Computational Linguistics 31(1): pp.25-69.

K. Crammer and Y. Singer. 2003. *Ultraconservative Online Algorithms for Multiclass Problems.* Journal of Machine Learning Research 3: pp.951-991.

S. Corston-Oliver and A. Aue. 2006. Dependency Parsing with Reference to Slovene, Spanish and Swedish. In *Proceedings of CoNLL-X.*

A. Culotta and J. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL 2004*.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2003. *Timbl: Tilburg memory based learner, version 5.0, reference guide.* Technical Report ILK 03-10, Tilburg University, ILK.

Y. Ding and M. Palmer. 2005. Machine Translation using Probabilistic Synchronous Dependency Insertion Grammars. In *Proceedings of ACL 2005.*

K. Hall and V. Novak. 2005. Corrective Modeling for Non-Projective Dependency Parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): pp. 313-330.

R. McDonald, F. Pereira, K. Ribarov and J. Hajič. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL 2006*.

T. Nakagawa, T. Kudo and Y. Matsumoto. 2002. Revision Learning and its Applications to Part-of-Speech Tagging. In *Proceedings of ACL 2002*.

J. Nilsson, J. Hall and J. Nivre. 2005. MAMBA Meets TIGER: Reconstructing a Swedish Treebank from Antiquity. In *Proceedings of the NODALIDA*.

J. Nivre and M. Scholz. 2004. Deterministic Dependency Parsing of English Text. In *Proceedings of COLING 2004*.

J. Nivre. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proceedings of CoNLL-X*.

F. Rosemblatt. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psych. Rev.*, 68: pp. 386-407.

G. Satta and E. Brill. 1995, Efficient Transformation-Based Parsing. In *Proceedings of ACL 1996*.

R. Snow, D. Jurafsky and Y. Ng 2005. Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Proceedings of NIPS 17*.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the 9th International Workshop on Parsing Technologies.*

Incremental Non-Projective Dependency Parsing

Joakim Nivre
Växjö University, School of Mathematics and Systems Engineering
Uppsala University, Department of Linguistics and Philology
`joakim.nivre@{msi.vxu.se,lingfil.uu.se}`

Abstract

An open issue in data-driven dependency parsing is how to handle non-projective dependencies, which seem to be required by linguistically adequate representations, but which pose problems in parsing with respect to both accuracy and efficiency. Using data from five different languages, we evaluate an incremental deterministic parser that derives non-projective dependency structures in $O(n^2)$ time, supported by SVM classifiers for predicting the next parser action. The experiments show that unrestricted non-projective parsing gives a significant improvement in accuracy, compared to a strictly projective baseline, with up to 35% error reduction, leading to state-of-the-art results for the given data sets. Moreover, by restricting the class of permissible structures to limited degrees of non-projectivity, the parsing time can be reduced by up to 50% without a significant decrease in accuracy.

1 Introduction

Data-driven dependency parsing has been shown to give accurate and efficient parsing for a wide range of languages, such as Japanese (Kudo and Matsumoto, 2002), English (Yamada and Matsumoto, 2003), Swedish (Nivre et al., 2004), Chinese (Cheng et al., 2004), and Czech (McDonald et al., 2005). Whereas most of the early approaches were limited to strictly *projective* dependency structures, where the projection of a syntactic head must be continuous, attention has recently shifted to the analysis of *non-projective* structures, which are required for linguistically adequate representations, especially in languages with free or flexible word order.

The most popular strategy for capturing non-projective structures in data-driven dependency parsing is to apply some kind of post-processing to the output of a strictly projective dependency parser, as in *pseudo-projective parsing* (Nivre and Nilsson, 2005), *corrective modeling* (Hall and Novák, 2005), or *approximate non-projective parsing* (McDonald and Pereira, 2006). And it is rare to find parsers that derive non-projective structures directly, the notable exception being the non-projective spanning tree parser proposed by McDonald et al. (2005).

There are essentially two arguments that have been advanced against using parsing algorithms that derive non-projective dependency structures directly. The first is that the added expressivity compromises efficiency, since the parsing problem for a grammar that allows arbitrary non-projective dependency structures has been shown to be $\mathcal{NP}$ complete (Neuhaus and Bröker, 1997). On the other hand, most data-driven approaches do not rely on grammars, and with a suitable factorization of dependency structures, it is possible to achieve parsing of unrestricted non-projective structures in $O(n^2)$ time, as shown by McDonald et al. (2005).

The second argument against non-projective dependency parsing comes from the observation that, even in languages with free or flexible word order,

Proceedings of NAACL HLT 2007, pages 396–403,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

most dependency structures are either projective or very nearly projective. This can be seen by considering data from treebanks, such as the Prague Dependency Treebank of Czech (Böhmová et al., 2003), the TIGER Treebank of German (Brants et al., 2002), or the Slovene Dependency Treebank (Džeroski et al., 2006), where the overall proportion of non-projective dependencies is only about 2% even though the proportion of sentences that contain some non-projective dependency is as high as 25%. This means that an approach that starts by deriving the best projective approximation of the correct dependency structure is likely to achieve high accuracy, while an approach that instead attempts to search the complete space of non-projective dependency structures runs the risk of finding structures that depart too much from the near-projective norm. Again, however, the results of McDonald et al. (2005) suggest that the latter risk is minimized if inductive learning is used to guide the search.

One way of improving efficiency, and potentially also accuracy, in non-projective dependency parsing is to restrict the search to a subclass of "mildly non-projective" structures. Nivre (2006) defines *degrees* of non-projectivity in terms of the maximum number of intervening constituents in the projection of a syntactic head and shows that limited degrees of non-projectivity give a much better fit with the linguistic data than strict projectivity, but also enables more efficient processing than unrestricted non-projectivity. However, the results presented by Nivre (2006) are all based on oracle parsing, which means that they only provide upper bounds on the accuracy that can be achieved.

In this paper, we investigate to what extent constraints on non-projective structures can improve accuracy and efficiency in practical parsing, using treebank-induced classifiers to predict the actions of a deterministic incremental parser. The parsing algorithm used belongs to the family of algorithms described by Covington (2001), and the classifiers are trained using support vector machines (SVM) (Vapnik, 1995). The system is evaluated using treebank data from five languages: Danish, Dutch, German, Portuguese, and Slovene.

The paper is structured as follows. Section 2 defines syntactic representations as labeled dependency graphs and introduces the notion of *degree* used to constrain the search. Section 3 describes the parsing algorithm, including modifications necessary to handle degrees of non-projectivity, and section 4 describes the data-driven prediction of parser actions, using history-based models and SVM classifiers. Section 5 presents the experimental setup, section 6 discusses the experimental results, and section 7 contains our conclusions.

2 Dependency Graphs

A dependency graph is a labeled directed graph, the nodes of which are indices corresponding to the tokens of a sentence. Formally:

Definition 1 Given a set R of dependency types (arc labels), a *dependency graph* for a sentence $x = (w_1, \ldots, w_n)$ is a labeled directed graph $G = (V, E, L)$, where:

1. $V = \{0, 1, 2, \ldots, n\}$
2. $E \subseteq V \times V$
3. $L : E \rightarrow R$

The set V of *nodes* (or *vertices*) is the set of non-negative integers up to and including n. This means that every token index i of the sentence is a node ($1 \leq i \leq n$) and that there is a special node 0, which will always be a root of the dependency graph. The set E of *arcs* (or *edges*) is a set of ordered pairs (i, j), where i and j are nodes. Since arcs are used to represent dependency relations, we will say that i is the *head* and j is the *dependent* of the arc (i, j). The function L assigns a dependency type (label) $r \in R$ to every arc $e \in E$. We use the notation $i \rightarrow j$ to mean that there is an arc connecting i and j (i.e., $(i, j) \in E$); we use the notation $i \xrightarrow{r} j$ if this arc is labeled r (i.e., $((i, j), r) \in L$); and we use the notation $i \rightarrow^* j$ and $i \leftrightarrow^* j$ for the reflexive and transitive closure of the arc relation E and the corresponding undirected relation, respectively.

Definition 2 A dependency graph G is *well-formed* if and only if:

1. The node 0 is a root, i.e., there is no node i such that $i \rightarrow 0$ (ROOT).
2. G is weakly connected, i.e., $i \leftrightarrow^* j$ for every pair of nodes i, j (CONNECTEDNESS).
3. Every node has at most one head, i.e., if $i \rightarrow j$ then there is no node k such that $k \neq i$ and $k \rightarrow j$ (SINGLE-HEAD).

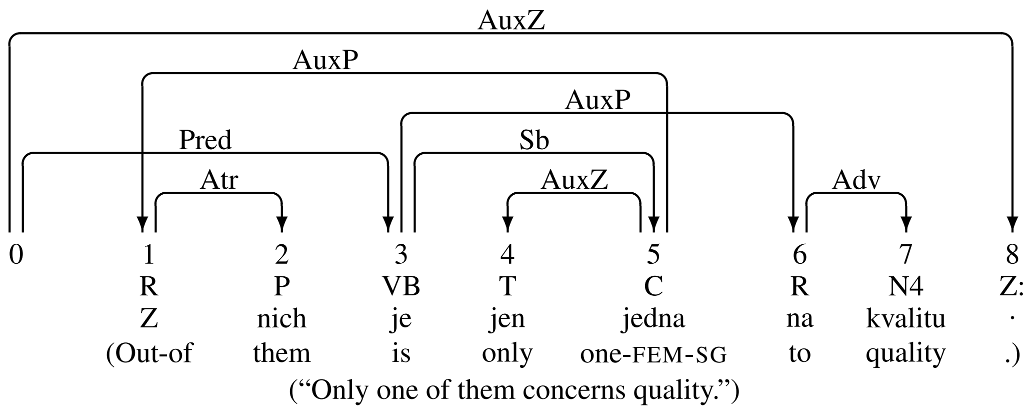


Figure 1: Dependency graph for Czech sentence from the Prague Dependency Treebank

The well-formedness conditions are independent in that none of them is entailed by any (combination) of the others, but they jointly entail that the graph is a tree rooted at the node 0. By way of example, figure 1 shows a Czech sentence from the Prague Dependency Treebank (Böhmová et al., 2003) with a well-formed dependency graph according to Definitions 1 and 2.

The constraints imposed on dependency graphs in Definition 2 are assumed in almost all versions of dependency grammar, especially in computational systems, and are sometimes complemented by a fourth constraint:

4. The graph G is projective, i.e., if $i \to j$ then $i \to^* k$, for every node k such that $i < k < j$ or $j < k < i$ (PROJECTIVITY).

Most theoretical formulations of dependency grammar regard projectivity as the norm but recognize the need for non-projective representations to capture non-local dependencies (Mel'čuk, 1988; Hudson, 1990). Finding a way of incorporating a suitably restricted notion of non-projectivity into practical parsing systems is therefore an important step towards a more adequate syntactic analysis, as discussed in the introduction of this paper.

In order to distinguish classes of dependency graphs that fall in between arbitrary non-projective and projective, Nivre (2006) introduces a notion of *degree* of non-projectivity, such that projective graphs have degree 0 while arbitrary non-projective graphs have unbounded degree.

Definition 3 Let $G = (V, E, L)$ be a well-formed dependency graph, let $G_{(i,j)}$ be the subgraph of G defined by $V_{(i,j)} = \{i, i+1, \ldots, j-1, j\}$, and let $\min(e)$ be the smallest and $\max(e)$ the largest element of an arc e in the linear order $<$:

1. The *degree* of an arc $e \in E$ is the number of connected components (i.e., weakly connected subgraphs) in $G_{(\min(e)+1,\max(e)-1)}$ that are not dominated by the head of e in $G_{(\min(e),\max(e))}$.
2. The *degree* of G is the maximum degree of any arc $e \in E$.

To exemplify the notion of degree, we note that the dependency graph in figure 1 has degree 1. The only non-projective arc in the graph is $(5, 1)$ and $G_{(2,4)}$ contains three connected components, each consisting of a single root node (2, 3, 4). Since exactly one of these, 3, is not dominated by 5 in $G_{(1,5)}$, the arc $(5, 1)$ has degree 1.

Nivre (2006) presents an empirical study, based on data from the Prague Dependency Treebank of Czech (Böhmová et al., 2003) and the Danish Dependency Treebank (Kromann, 2003), showing that more than 99.5% of all sentences occurring in the two treebanks have a dependency graph with a maximum degree of 2; about 98% have a maximum degree of 1; but only 77% in the Czech data and 85% in the Danish data have degree 0 (which is equivalent to assuming PROJECTIVITY). This suggests that limited degrees of non-projectivity may allow a parser to capture a larger class of naturally occurring syntactic structures, while still constraining the search to a proper subclass of all possible structures.[1]

[1]Alternative notions of mildly non-projective dependency structures are explored in Kuhlmann and Nivre (2006).

3 Parsing Algorithm

Covington (2001) describes a parsing strategy for dependency representations that has been known since the 1960s but not presented in the literature. The left-to-right (or incremental) version of this strategy can be formulated in the following way:

```
PARSE($x = (w_1, \ldots, w_n)$)
1  for $j = 1$ up to $n$
2      for $i = j - 1$ down to $0$
3          LINK($i$, $j$)
```

LINK(i, j) is a nondeterministic operation that adds the arc $i \rightarrow j$ (with some label), adds the arc $j \rightarrow i$ (with some label), or does nothing at all. In this way, the algorithm builds a graph by systematically trying to link every pair of nodes (i, j) $(i < j)$. We assume that LINK(i,j) respects the ROOT and SINGLE-HEAD constraints and that it does not introduce cycles into the graph, i.e., it adds an arc $i \rightarrow j$ only if $j \neq 0$, there is no $k \neq i$ such that $k \rightarrow j$, and it is not the case that $j \rightarrow^* i$. Given these constraints, the graph G given at termination can always be turned into a well-formed dependency graph by adding arcs from the root 0 to any root node in $\{1, \ldots, n\}$.

Assuming that LINK(i, j) can be performed in some constant time c, the running time of the algorithm is $\sum_{i=1}^{n} c(i-1) = c(\frac{n^2}{2} - \frac{n}{2})$, which in terms of asymptotic complexity is $O(n^2)$. Checking ROOT and SINGLE-HEAD in constant time is easy, but in order to prevent cycles we need to be able to find, for any node k, the root of the connected component to which k belongs in the partially built graph. This problem can be solved efficiently using standard techniques for disjoint sets, including path compression and union by rank, which guarantee that the necessary checks can be performed in average constant time (Cormen et al., 1990).

In the experiments reported in this paper, we modify the basic algorithm by making the performance of LINK(i, j) conditional on the arcs (i, j) and (j, i) being permissible under different degree constraints:

```
PARSE($x = (w_1, \ldots, w_n)$, $d$)
1  for $j = 1$ up to $n$
2      for $i = j - 1$ down to $0$
3          if PERMISSIBLE($i$, $j$, $d$)
4              LINK($i$, $j$)
```

The function PERMISSIBLE(i, j, d) returns **true** if and only if $i \rightarrow j$ and $j \rightarrow i$ have a degree less than or equal to d given the partially built graph G. Setting $d = 0$ gives strictly projective parsing, while $d = \infty$ corresponds to unrestricted non-projective parsing. With low values of d, we will reduce the number of calls to LINK(i, j), which will reduce the overall parsing time provided that the time required to compute PERMISSIBLE(i, j, d) is insignificant compared to the time needed for LINK(i, j). This is typically the case in data-driven systems, where LINK(i, j) requires a call to a trained classifier, while PERMISSIBLE(i, j, d) only needs access to the partially built graph G.[2]

4 History-Based Parsing

History-based parsing uses features of the parsing history to predict the next parser action (Black et al., 1992). In the current setup, this involves using features of the partially built dependency graph G and the input $x = (w_1, \ldots, w_n)$ to predict the outcome of the nondeterministic LINK(i, j) operation. Given that we use a deterministic parsing strategy, this reduces to a pure classification problem.

Let $\Phi(i, j, G) = (\phi_1, \ldots, \phi_m)$ be a feature vector representation of the parser history at the time of performing LINK(i, j). The task of the history-based classifier is then to map $\Phi(i, j, G)$ to one of the following actions:

1. Add the arc $i \xrightarrow{r} j$ (for some $r \in R$).
2. Add the arc $j \xrightarrow{r} i$ (for some $r \in R$).
3. Do nothing.

Training data for the classifier can be generated by running the parser on a sample of treebank data, using the gold standard dependency graph as an oracle to predict LINK(i, j) and constructing one training instance $(\Phi(i, j, G), a)$ for each performance of LINK(i, j) with outcome a.

The features in $\Phi(i, j, G) = (\phi_1, \ldots, \phi_m)$ can be arbitrary features of the input x and the partially built graph G but will in the experiments below be restricted to linguistic attributes of input tokens, including their dependency types according to G.

[2]Checking PERMISSIBLE(i, j, d), again requires finding the roots of connected components and can therefore be done in average constant time.

Language	Tok	Sen	T/S	Lem	CPoS	PoS	MSF	Dep	NPT	NPS
Danish	94	5.2	18.2	no	10	24	47	52	1.0	15.6
Dutch	195	13.3	14.6	yes	13	302	81	26	5.4	36.4
German	700	39.2	17.8	no	52	52	0	46	2.3	27.8
Portuguese	207	9.1	22.8	yes	15	21	146	55	1.3	18.9
Slovene	29	1.5	18.7	yes	11	28	51	25	1.9	22.2

Table 1: Data sets; Tok = number of tokens (*1000); Sen = number of sentences (*1000); T/S = tokens per sentence (mean); Lem = lemmatization present; CPoS = number of coarse-grained part-of-speech tags; PoS = number of (fine-grained) part-of-speech tags; MSF = number of morphosyntactic features (split into atoms); Dep = number of dependency types; NPT = proportion of non-projective dependencies/tokens (%); NPS = proportion of non-projective dependency graphs/sentences (%)

The history-based classifier can be trained with any of the available supervised methods for function approximation, but in the experiments below we will rely on SVM, which has previously shown good performance for this kind of task (Kudo and Matsumoto, 2002; Yamada and Matsumoto, 2003).

5 Experimental Setup

The purpose of the experiments is twofold. First, we want to investigate whether allowing non-projective structures to be derived incrementally can improve parsing accuracy compared to a strictly projective baseline. Secondly, we want to examine whether restricting the degree of non-projectivity can improve efficiency compared to an unrestricted non-projective baseline. In order to investigate both these issues, we have trained one non-projective parser for each language, allowing arbitrary non-projective structures as found in the treebanks during training, but applying different constraints during parsing:

1. Non-projective ($d = \infty$)
2. Max degree 2 ($d = 2$)
3. Max degree 1 ($d = 1$)

These three versions of the non-projective parser are compared to a strictly projective parser ($d = 0$), which uses the same parsing algorithm but only considers projective arcs in both training and testing.[3]

The experiments are based on treebank data from five languages: the Danish Dependency Treebank (Kromann, 2003), the Alpino Treebank of Dutch (van der Beek et al., 2002), the TIGER Treebank of German (Brants et al., 2002), the Floresta Sintáctica of Portuguese (Afonso et al., 2002), and the Slovene Dependency Treebank (Džeroski et al., 2006).[4] The data sets used are the training sets from the CoNLL-X Shared Task on multilingual dependency parsing (Buchholz and Marsi, 2006), with 20% of the data reserved for testing using a pseudo-random split. Table 1 gives an overview of the five data sets, showing the number of tokens and sentences, the presence of different kinds of linguistic annotation, and the amount of non-projectivity.

The features used in the history-based model for all languages include the following core set of 20 features, where i and j are the tokens about to be linked and the *context stack* is a stack of root nodes k in $G_{(i+1,j-1)}$, added from right to left (i.e., with the top node being closest to i):

1. Word form: i, j, $j+1$, $h(i)$.
2. Lemma (if available): i.
3. Part-of-speech: $i-1$, i, j, $j+1$, $j+2$, k, $k-1$.
4. Coarse part-of-speech (if available): i, j, k.
5. Morphosyntactic features (if available): i, j.
6. Dependency type: i, j, $l(i)$, $l(j)$, $r(i)$.

In the specification of features, we use k and $k-1$ to refer to the two topmost tokens on the context stack, and we use $h(\alpha)$, $l(\alpha)$ and $r(\alpha)$ to refer to the *head*,

[3]An alternative would have been to train all parsers on non-projective data, or restrict the training data for each parser according to its parsing restriction. Preliminary experiments showed that the setup used here gave the best performance for all parsers involved.

[4]This set does not include the Prague Dependency Treebank of Czech (Böhmová et al., 2003), one of the most widely used treebanks in studies of non-projective parsing. The reason is that the sheer size of this data set makes extensive experiments using SVM learning extremely time consuming. The work on Czech was therefore initially postponed but is now ongoing.

	Danish		**Dutch**		**German**		**Portuguese**		**Slovene**	
Constraint	**AS**	**ER**	**AS**	**ER**	**AS**	**ER**	**AS**	**ER**	**AS**	**ER**
Non-projective	88.13	8.34	86.79	36.18	89.78	21.51	90.59	11.39	76.52	6.83
Max degree 2	88.08	7.95	86.15	33.09	89.74	21.20	90.58	11.30	76.48	6.67
Max degree 1	88.00	7.33	85.12	28.12	89.49	19.28	90.48	10.36	76.40	6.35
Projective	87.05	–	79.30	–	86.98	–	89.38	–	74.80	–

Table 2: Parsing accuracy; AS = attachment score; ER = error reduction w.r.t. projective baseline (%)

the *leftmost dependent* and the *rightmost dependent* of a token α in the partially built dependency graph.[5] In addition to the core set of features, the model for each language has been augmented with a small number of additional features, which have proven useful in previous experiments with the same data set. The maximum number of features used is 28 (Danish); the minimum number is 23 (German).

The history-based classifiers have been trained using SVM learning, which combines a maximum margin strategy with the use of kernel functions to map the original feature space to a higher-dimensional space. More specifically, we use LIBSVM (Chang and Lin, 2001) with a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$. We use the built-in one-versus-one strategy for multi-class classification and convert symbolic features to numerical features using the standard technique of binarization.

Parsing accuracy is measured by the unlabeled attachment score (AS), i.e., the proportion of words that are assigned the correct head (not counting punctuation). Although the parsers do derive *labeled* dependency graphs, we concentrate on the graph structure here, since this is what is concerned in the distinction between projective and non-projective dependency graphs. Efficiency is evaluated by reporting the parsing time (PT), i.e., the time required to parse the respective test sets. Since both training sets and test sets vary considerably in size between languages, we are primarily interested in the relative differences for parsers applied to the same language. Experiments have been performed on a Sun-Blade 2000 with one 1.2GHz UltraSPARC-III processor and 2GB of memory.

[5]The lack of symmetry in the feature set reflects the asymmetry in the partially built graph G, where, e.g., only i can have dependents to the right at decision time. This explains why there are more features defined in terms of graph structure for i and more features defined in terms of string context for j.

6 Results and Discussion

Table 2 shows the parsing accuracy of the non-projective parser with different maximum degrees, both the raw attachment scores and the amount of error reduction with respect to the baseline parser. Our first observation is that the non-projective parser invariably achieves higher accuracy than the projective baseline, with differences that are statistically significant across the board (using McNemar's test). The amount of error reduction varies between languages and seems to depend primarily on the frequency of non-projective structures, which is not surprising. Thus, for Dutch and German, the two languages with the highest proportion of non-projective structures, the best error reduction is over 35% and over 20%, respectively. However, there seems to be a sparse data effect in that Slovene, which has the smallest training data set, has the smallest error reduction despite having more non-projective structures than Danish and Portuguese.

Our second observation is that the highest score is always obtained with an unbounded degree of non-projectivity during parsing. This seems to corroborate the results obtained by McDonald et al. (2005) with a different parsing method, showing that the use of inductive learning to guide the search during parsing eliminates the potentially harmful effect of increasing the size of the search space. Although the differences between different degrees of non-projectivity are not statistically significant for the current data sets,[6] the remarkable consistency across languages suggests that they are nevertheless genuine. In either case, however, they must be considered marginal, except possibly for Dutch, which leads to our third and final observation about accu-

[6]The only exception is the difference between a maximum degree of 1 and unrestricted non-projective for Dutch, which is significant according to McNemar's test with $\alpha = .05$.

	Danish		Dutch		German		Portuguese		Slovene	
Constraint	**PT**	**TR**	**PT**	**TR**	**PT**	**TR**	**PT**	**TR**	**PT**	**TR**
Non-projective	426	–	3791	–	24454	–	3708	–	204	–
Max degree 2	395	7.29	2068	45.46	17903	26.79	3004	18.99	130	36.39
Max degree 1	346	18.72	1695	55.28	13079	46.52	2446	34.04	108	47.05
Projective	211	50.53	784	79.32	7362	69.90	1389	62.55	429	79.00

Table 3: Parsing time; PT = parsing time (s); TR = time reduction w.r.t. non-projective baseline (%)

System	**Danish**	**Dutch**	**German**	**Portuguese**	**Slovene**
CoNLL-X McDonald et al.	84.79	79.19	87.34	86.82	73.44
CoNLL-X Nivre et al.	84.77	78.59	85.82	87.60	70.30
Incremental non-projective	84.85	77.91	85.90	87.12	70.86

Table 4: Related work (labeled attachment score)

racy, namely that restricting the maximum degree of non-projectivity to 2 or 1 has a very marginal effect on accuracy and is always significantly better than the projective baseline.

Turning next to efficiency, table 3 shows the parsing time for the different parsers across the five languages. Our first observation here is that the parsing time can be reduced by restricting the degree of non-projectivity during parsing, thus corroborating the claim that the running time of the history-based classifier dominates the overall parsing time. As expected, the largest reduction is obtained with the strictly projective parser, but here we must also take into account that the training data set is smaller (because of the restriction to projective potential links), which improves the average running time of the history-based classifier in itself. Our second observation is that the amount of reduction in parsing time seems to be roughly related to the amount of non-projectivity, with a reduction of about 50% at a max degree of 1 for the languages where more than 20% of all sentences are non-projective (Dutch, German, Slovene) but significantly smaller for Portuguese and especially for Danish. On the whole, however, the reduction in parsing time with limited degrees of non-projectivity is substantial, especially considering the very marginal drop in accuracy.

In order to compare the performance to the state of the art in dependency parsing, we have retrained the non-projective parser on the entire training data set for each language and evaluated it on the final test set from the CoNLL-X shared task (Buchholz and Marsi, 2006). Thus, table 4 shows *labeled* attachment scores, the main evaluation metric used in the shared task, in comparison to the two highest scoring systems from the original evaluation (McDonald et al., 2006; Nivre et al., 2006). The incremental non-projective parser has the best reported score for Danish and outperforms at least one of the other two systems for four languages out of five, although most of the differences are probably too small to be statistically significant. But whereas the spanning tree parser of McDonald et al. (2006) and the pseudo-projective parser of Nivre et al. (2006) achieve this performance only with special pre- or post-processing,[7] the approach presented here derives a labeled non-projective graph in a single incremental process and hence at least has the advantage of simplicity. Moreover, it has better time complexity than the approximate second-order spanning tree parsing of McDonald et al. (2006), which has exponential complexity in the worst case (although this does not appear to be a problem in practice).

7 Conclusion

In this paper, we have investigated a data-driven approach to dependency parsing that combines a deterministic incremental parsing algorithm with history-based SVM classifiers for predicting the next parser action. We have shown that, for languages with a

[7]McDonald et al. (2006) use post-processing for non-projective dependencies and for labeling. Nivre et al. (2006) use pre-processing of training data and post-processing of parser output to recover non-projective dependencies.

non-negligible proportion of non-projective structures, parsing accuracy can be improved significantly by allowing non-projective structures to be derived. We have also shown that the parsing time can be reduced substantially, with only a marginal loss in accuracy, by limiting the degree of non-projectivity allowed during parsing. A comparison with results from the CoNLL-X shared task shows that the parsing accuracy is comparable to that of the best available systems, which means that incremental non-projective dependency parsing is a viable alternative to approaches based on post-processing of projective approximations.

Acknowledgments

The research presented in this paper was partially supported by a grant from the Swedish Research Council. I want to thank Johan Hall and Jens Nilsson for their contributions to MaltParser, which was used to perform the experiments. I am also grateful to three anonymous reviewers for finding important errors in the preliminary version and for suggesting several other improvements for the final version.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. “Floresta sintá(c)tica”: a treebank for Portuguese. In *Proc. of LREC*, 1698–1703.

E. Black, F. Jelinek, J. D. Lafferty, D. M. Magerman, R. L. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of the DARPA Speech and Natural Language Workshop*, 31–37.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeillé, ed., *Treebanks: Building and Using Parsed Corpora*, chapter 7. Kluwer, Dordrecht.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of TLT*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*, 149–164.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Y. Cheng, M. Asahara, and Y. Matsumoto. 2004. Deterministic dependency structure analyzer for Chinese. In *Proc. of IJCNLP*, 500–508.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to Algorithms*. MIT Press.

M. A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. of the Annual ACM Southeast Conference*, 95–102.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of LREC*.

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proc. of IWPT*, 42–52

R. A. Hudson. 1990. *English Word Grammar*. Blackwell.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of TLT*.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL*, 63–69.

M. Kuhlmann and J. Nivre. 2006. Mildly non-projective dependency structures. In *Proc. of COLING-ACL, Posters*, 507–514.

R. McDonald and F-. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, 81–88.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*, 523–530.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of CoNLL*, 216–220.

I. Mel’čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

P. Neuhaus and N. Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proc. of ACL-EACL*, 337–343.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*, 99–106.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of CoNLL*, 49–56.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of CoNLL*, 221–225.

J. Nivre. 2006. Constraints on non-projective dependency graphs. In *Proc. of EACL*, 73–80.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

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

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*, 195–206.

Improved Inference for Unlexicalized Parsing

Slav Petrov and Dan Klein
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{petrov,klein}@eecs.berkeley.edu

Abstract

We present several improvements to unlexicalized parsing with hierarchically state-split PCFGs. First, we present a novel coarse-to-fine method in which a grammar's own hierarchical projections are used for incremental pruning, including a method for efficiently computing projections of a grammar without a treebank. In our experiments, hierarchical pruning greatly accelerates parsing with no loss in empirical accuracy. Second, we compare various inference procedures for state-split PCFGs from the standpoint of risk minimization, paying particular attention to their practical tradeoffs. Finally, we present multilingual experiments which show that parsing with hierarchical state-splitting is fast and accurate in multiple languages and domains, even without any language-specific tuning.

1 Introduction

Treebank parsing comprises two problems: *learning*, in which we must select a model given a treebank, and *inference*, in which we must select a parse for a sentence given the learned model. Previous work has shown that high-quality unlexicalized PCFGs can be learned from a treebank, either by manual annotation (Klein and Manning, 2003) or automatic state splitting (Matsuzaki et al., 2005; Petrov et al., 2006). In particular, we demonstrated in Petrov et al. (2006) that a hierarchically split PCFG could exceed the accuracy of lexicalized PCFGs (Collins, 1999; Charniak and Johnson, 2005). However, many questions about inference with such split PCFGs remain open. In this work, we present

1. an effective method for pruning in split PCFGs
2. a comparison of objective functions for inference in split PCFGs,
3. experiments on automatic splitting for languages other than English.

In Sec. 3, we present a novel coarse-to-fine processing scheme for hierarchically split PCFGs. Our method considers the splitting history of the final grammar, projecting it onto its increasingly refined prior stages. For any projection of a grammar, we give a new method for efficiently estimating the projection's parameters from the source PCFG itself (rather than a treebank), using techniques for infinite tree distributions (Corazza and Satta, 2006) and iterated fixpoint equations. We then parse with each refinement, in sequence, much along the lines of Charniak et al. (2006), except with much more complex and automatically derived intermediate grammars. Thresholds are automatically tuned on held-out data, and the final system parses up to 100 times faster than the baseline PCFG parser, with no loss in test set accuracy.

In Sec. 4, we consider the well-known issue of inference objectives in split PCFGs. As in many model families (Steedman, 2000; Vijay-Shanker and Joshi, 1985), split PCFGs have a derivation / parse distinction. The split PCFG directly describes a generative model over derivations, but evaluation is sensitive only to the coarser treebank symbols. While the most probable parse problem is NP-complete (Sima'an, 1992), several approximate methods exist, including n-best reranking by parse likelihood, the labeled bracket algorithm of Goodman (1996), and a variational approximation introduced in Matsuzaki et al. (2005). We present experiments which explicitly minimize various evaluation risks over a candidate set using samples from the split PCFG, and relate those conditions to the existing non-sampling algorithms. We demonstrate that n-best reranking according to likelihood is superior for exact match, and that the non-reranking methods are superior for maximizing F_1. A specific contribution is to discuss the role of unary productions, which previous work has glossed over, but which is important in understanding why the various methods work as they do.

Finally, in Sec. 5, we learn state-split PCFGs for German and Chinese and examine out-of-domain performance for English. The learned grammars are compact and parsing is very quick in our multi-stage scheme. These grammars produce the highest test set parsing figures that we are aware of in each language, except for English for which non-local methods such as feature-based discriminative reranking are available (Charniak and Johnson, 2005).

2 Hierarchically Split PCFGs

We consider PCFG grammars which are derived from a raw treebank as in Petrov et al. (2006): A simple X-bar grammar is created by binarizing the treebank trees. We refer to this grammar as G_0. From this starting point, we iteratively refine the grammar in stages, as illustrated in Fig. 1. In each stage, all symbols are split in two, for example *DT* might become *DT-1* and *DT-2*. The refined grammar is estimated using a variant of the forward-backward algorithm (Matsuzaki et al., 2005). After a splitting stage, many splits are rolled back based on (an approximation to) their likelihood gain. This procedure gives an ontogeny of grammars G_i, where $G = G_n$ is the final grammar. Empirically, the gains on the English Penn treebank level off after 6 rounds. In Petrov et al. (2006), some simple smoothing is also shown to be effective. It is interesting to note that these grammars capture many of the "structural zeros" described by Mohri and Roark (2006) and pruning rules with probability below e^{-10} reduces the grammar size drastically without influencing parsing performance. Some of our methods and conclusions are relevant to all state-split grammars, such as Klein and Manning (2003) or Dreyer and Eisner (2006), while others apply most directly to the hierarchical case.

3 Search

When working with large grammars, it is standard to prune the search space in some way. In the case of lexicalized grammars, the unpruned chart often will not even fit in memory for long sentences. Several proven techniques exist. Collins (1999) combines a punctuation rule which eliminates many spans entirely, and then uses span-synchronous beams to prune in a bottom-up fashion. Charniak et al. (1998) introduces best-first parsing, in which a figure-of-merit prioritizes agenda processing. Most relevant to our work is Charniak and Johnson (2005) which uses a *pre-parse* phase to rapidly parse with a very coarse, unlexicalized treebank grammar. Any item X:$[i, j]$ with sufficiently low posterior probability in the pre-parse triggers the pruning of its lexical variants in a subsequent full parse.

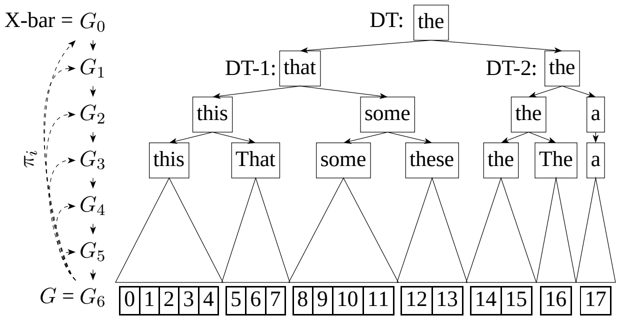


Figure 1: Hierarchical refinement proceeds top-down while projection recovers coarser grammars. The top word for the first refinements of the determiner tag (DT) is shown on the right.

3.1 Coarse-to-Fine Approaches

Charniak et al. (2006) introduces *multi-level coarse-to-fine* parsing, which extends the basic pre-parsing idea by adding more rounds of pruning. In their work, the extra pruning was with grammars even coarser than the raw treebank grammar, such as a grammar in which all nonterminals are collapsed. We propose a novel multi-stage coarse-to-fine method which is particularly natural for our hierarchically split grammar, but which is, in principle, applicable to any grammar. As in Charniak et al. (2006), we construct a sequence of increasingly refined grammars, reparsing with each refinement. The contributions of our method are that we derive sequences of refinements in a new way (Sec. 3.2), we consider refinements which are themselves complex, and, because our full grammar is not impossible to parse with, we automatically tune the pruning thresholds on held-out data.

3.2 Projection

In our method, which we call *hierarchical coarse-to-fine* parsing, we consider a sequence of PCFGs $G_0, G_1, \dots G_n = G$, where each G_i is a refinement of the preceding grammar G_{i-1} and G is the full grammar of interest. Each grammar G_i is related to $G = G_n$ by a *projection* $\pi_{n\to i}$ or π_i for brevity. A projection is a map from the non-terminal (including pre-terminal) symbols of G onto a reduced domain. A projection of grammar symbols induces a projection of rules and therefore entire non-weighted grammars (see Fig. 1).

In our case, we also require the projections to be sequentially compatible, so that $\pi_{i\to j} = \pi_{k\to j} \circ \pi_{i\to k}$. That is, each projection is itself a coarsening of the previous projections. In particular, we take the projection $\pi_{i\to j}$ to be the map that collapses split symbols in round i to their earlier identities in round j.

It is straightforward to take a projection π and map a CFG G to its induced projection $\pi(G)$. What is less obvious is how the probabilities associated with the rules of G should be mapped. In the case where $\pi(G)$ is more coarse than the treebank originally used to train G, and when that treebank is available, it is easy to project the treebank and directly estimate, say, the maximum-likelihood parameters for $\pi(G)$. This is the approach taken by Charniak et al. (2006), where they estimate what in our terms are projections of the raw treebank grammar from the treebank itself.

However, treebank estimation has several limitations. First, the treebank used to train G may not be available. Second, if the grammar G is heavily smoothed or otherwise regularized, its own distribution over trees may be far from that of the treebank. Third, the meanings of the split states can and do drift between splitting stages. Fourth, and most importantly, we may wish to project grammars for which treebank estimation is problematic, for example, grammars which are more refined than the observed treebank grammars. Our method effectively avoids all of these problems by rebuilding and refitting the pruning grammars on the fly from the final grammar.

3.2.1 Estimating Projected Grammars

Fortunately, there is a well worked-out notion of estimating a grammar from an infinite distribution over trees (Corazza and Satta, 2006). In particular, we can estimate parameters for a projected grammar $\pi(G)$ from the tree distribution induced by G (which can itself be estimated in any manner). The earliest work that we are aware of on estimating models from models in this way is that of Nederhof (2005), who considers the case of learning language models from other language models. Corazza and Satta (2006) extend these methods to the case of PCFGs and tree distributions.

The generalization of maximum likelihood estimation is to find the estimates for $\pi(G)$ with minimum KL divergence from the tree distribution induced by G. Since $\pi(G)$ is a grammar over coarser symbols, we fit $\pi(G)$ to the distribution G induces over π-projected trees: $P(\pi(T)|G)$. The proofs of the general case are given in Corazza and Satta (2006), but the resulting procedure is quite intuitive.

Given a (fully observed) treebank, the maximum-likelihood estimate for the probability of a rule $X \to YZ$ would simply be the ratio of the count of X to the count of the configuration $X \to YZ$. If we wish to find the estimate which has minimum divergence to an infinite distribution $P(T)$, we use the same formula, but the counts become expected counts:

$$P(X \to YZ) = \frac{E_{P(T)}[X \to YZ]}{E_{P(T)}[X]}$$

with unaries estimated similarly. In our specific case, $X, Y,$ and Z are symbols in $\pi(G)$, and the expectations are taken over G's distribution of π-projected trees, $P(\pi(T)|G)$. We give two practical methods for obtaining these expectations below.

3.2.2 Calculating Projected Expectations

Concretely, we can now estimate the minimum divergence parameters of $\pi(G)$ for any projection π and PCFG G if we can calculate the expectations of the projected symbols and rules according to $P(\pi(T)|G)$. The simplest option is to sample trees T from G, project the samples, and take average counts off of these samples. In the limit, the counts will converge to the desired expectations, provided the grammar is proper. However, we can exploit the structure of our projections to obtain the desired expectations much more simply and efficiently.

First, consider the problem of calculating the expected counts of a symbol X in a tree distribution given by a grammar G, ignoring the issue of projection. These expected counts obey the following one-step equations (assuming a unique *root* symbol):

$$c(root) = 1$$

$$c(X) = \sum_{Y \to \alpha X \beta} P(\alpha X \beta | Y) c(Y)$$

Here, α, β, or both can be empty, and a rule $X \rightarrow \gamma$ appears in the sum once for each X it contains. In principle, this linear system can be solved in any way.[1] In our experiments, we solve this system iteratively, with the following recurrences:

$$c_0(X) \leftarrow \begin{cases} 1 & \text{if } X = root \\ 0 & \text{otherwise} \end{cases}$$

$$c_{i+1}(X) \leftarrow \sum_{Y \rightarrow \alpha X \beta} P(\alpha X \beta | Y) c_i(Y)$$

Note that, as in other iterative fixpoint methods, such as policy evaluation for Markov decision processes (Sutton and Barto, 1998), the quantities $c_k(X)$ have a useful interpretation as the expected counts ignoring nodes deeper than depth k (i.e. the roots are all the root symbol, so $c_0(root) = 1$). In our experiments this method converged within around 25 iterations; this is unsurprising, since the treebank contains few nodes deeper than 25 and our base grammar G seems to have captured this property.

Once we have the expected counts of symbols in G, the expected counts of their projections $X' = \pi(X)$ according to $P(\pi(T)|G)$ are given by $c(X') = \sum_{X:\pi(X)=X'} c(X)$. Rules can be estimated directly using similar recurrences, or given by one-step equations:

$$c(X \rightarrow \gamma) = c(X) P(\gamma | X)$$

This process very rapidly computes the estimates for a projection of a grammar (i.e. in a few seconds for our largest grammars), and is done once during initialization of the parser.

3.2.3 Hierarchical Projections

Recall that our final state-split grammars G come, by their construction process, with an ontogeny of grammars G_i where each grammar is a (partial) splitting of the preceding one. This gives us a natural chain of projections $\pi_{i \rightarrow j}$ which projects backwards along this ontogeny of grammars (see Fig. 1). Of course, training also gives us parameters for the grammars, but only the chain of projections is needed. Note that the projected estimates need not (and in general will not) recover the original parameters exactly, nor would we want them to. Instead they take into account any smoothing, substate drift, and so on which occurred by the final grammar.

Starting from the base grammar, we run the projection process for each stage in the sequence, calculating π_i (chained incremental projections would also be possible). For the remainder of the paper, except where noted otherwise, all coarser grammars' estimates are these reconstructions, rather than those originally learned.

3.3 Experiments

As demonstrated by Charniak et al. (2006) parsing times can be greatly reduced by pruning chart items that have low posterior probability under a simpler grammar. Charniak et al. (2006) pre-parse with a sequence of grammars which are coarser than (parent-annotated) treebank grammars. However, we also work with grammars which are already heavily split, up to half as split as the final grammar, because we found the computational cost for parsing with the simple X-bar grammar to be insignificant compared to the costs for parsing with more refined grammars.

For a final grammar $G = G_n$, we compute estimates for the n projections $G_{n-1}, \ldots, G_0$ =X-Bar, where $G_i = \pi_i(G)$ as described in the previous section. Additionally we project to a grammar G_{-1} in which all nonterminals, except for the preterminals, have been collapsed. During parsing, we start of by exhaustively computing the inside/outside scores with G_{-1}. At each stage, chart items with low posterior probability are removed from the chart, and we proceed to compute inside/outside scores with the next, more refined grammar, using the projections $\pi_{i \rightarrow i-1}$ to map between symbols in G_i and G_{i-1}. In each pass, we skip chart items whose projection into the previous stage had a probability below a stage-specific threshold, until we reach $G = G_n$ (after seven passes in our case). For G, we do not prune but instead return the minimum risk tree, as will be described in Sec. 4.

Fig. 2 shows the (unlabeled) bracket posteriors after each pass and demonstrates that most constructions can be ruled out by the simpler grammars, greatly reducing the amount of computation for the following passes. The pruning thresholds were empirically determined on a held out set by computing

[1]Whether or not the system has solutions depends on the parameters of the grammar. In particular, G may be improper, though the results of Chi (1999) imply that G will be proper if it is the maximum-likelihood estimate of a finite treebank.

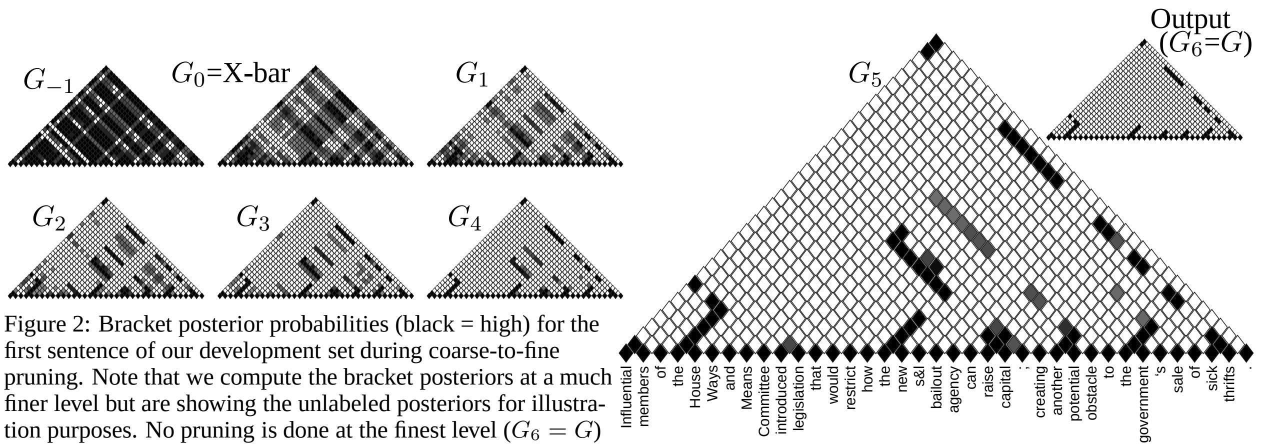


Figure 2: Bracket posterior probabilities (black = high) for the first sentence of our development set during coarse-to-fine pruning. Note that we compute the bracket posteriors at a much finer level but are showing the unlabeled posteriors for illustration purposes. No pruning is done at the finest level ($G_6 = G$) but the minimum risk tree is returned instead.

the most likely tree under G directly (without pruning) and then setting the highest pruning threshold for each stage that would not prune the optimal tree. This setting also caused no search errors on the test set. We found our projected grammar estimates to be at least equally well suited for pruning as the original grammar estimates which were learned during the hierarchical training. Tab. 1 shows the tremendous reduction in parsing time (all times are cumulative) and gives an overview over grammar sizes and parsing accuracies. In particular, in our Java implementation on a 3GHz processor, it is possible to parse the 1578 development set sentences (of length 40 or less) in less than 1200 seconds with an F$_1$ of 91.2% (no search errors), or, by pruning more, in 680 seconds at 91.1%. For comparison, the Feb. 2006 release of the Charniak and Johnson (2005) parser runs in 1150 seconds on the same machine with an F$_1$ of 90.7%.

4 Objective Functions for Parsing

A split PCFG is a grammar G over symbols of the form X-k where X is an evaluation symbol (such as *NP*) and k is some indicator of a subcategory, such as a parent annotation. G induces a *derivation distribution* $P(T|G)$ over trees T labeled with split symbols. This distribution in turn induces a *parse distribution* $P(T'|G) = P(\pi(T)|G)$ over (projected) trees with unsplit evaluation symbols, where $P(T'|G) = \sum_{T:T'=\pi(T)} P(T|G)$. We now have several choices of how to select a tree given these posterior distributions over trees. In this section, we present experiments with the various options and explicitly relate them to parse risk minimization (Titov and Henderson, 2006).

	G$_0$	G$_2$	G$_4$	G$_6$
Nonterminals	98	219	498	1140
Rules	3,700	19,600	126,100	531,200
No pruning	52 min	99 min	288 min	1612 min
X-bar pruning	8 min	14 min	30 min	111 min
C-to-F (no loss)	6 min	12 min	16 min	20 min
F$_1$ for above	64.8	85.2	89.7	91.2
C-to-F (lossy)	6 min	8 min	9 min	11 min
F$_1$ for above	64.3	84.7	89.4	91.1

Table 1: Grammar sizes, parsing times and accuracies for hierarchically split PCFGs with and without hierarchical coarse-to-fine parsing on our development set (1578 sentences with 40 or less words from section 22 of the Penn Treebank). For comparison the parser of Charniak and Johnson (2005) has an accuracy of F$_1$=90.7 and runs in 19 min on this set.

The decision-theoretic approach to parsing would be to select the parse tree which minimizes our expected loss according to our beliefs:

$$T_P^* = \underset{T_P}{\operatorname{argmin}} \sum_{T_T} P(T_T|w, G) L(T_P, T_T)$$

where T_T and T_P are "true" and predicted parse trees. Here, our loss is described by the function L whose first argument is the predicted parse tree and the second is the gold parse tree. Reasonable candidates for L include zero-one loss (exact match), precision, recall, F$_1$ (specifically EVALB here), and so on. Of course, the naive version of this process is intractable: we have to loop over all (pairs of) possible parses. Additionally, it requires parse likelihoods $P(T_P|w, G)$, which are tractable, but not trivial, to compute for split models. There are two options: limit the predictions to a small candidate set or choose methods for which dynamic programs exist.

For arbitrary loss functions, we can approximate the minimum-risk procedure by taking the min over only a set of *candidate parses* T_P. In some cases, each parse's expected risk can be evaluated in closed

Rule score: $r(\text{A} \to \text{B C}, i, k, j) = \sum_x \sum_y \sum_z \text{P}_{\text{OUT}}(A_x, i, j)\text{P}(\text{A}_x \to \text{B}_y \text{ C}_z)\text{P}_{\text{IN}}(B_y, i, k)\text{P}_{\text{IN}}(C_y, k, j)$

VARIATIONAL: $q(\text{A} \to \text{B C}, i, k, j) = \frac{r(\text{A} \to \text{B C}, i, k, j)}{\sum_x \text{P}_{\text{OUT}}(A_x, i, j)\text{P}_{\text{IN}}(A_x, i, j)}$ $\quad T_G = \text{argmax}_T \prod_{e \in T} q(e)$

MAX-RULE-SUM: $q(\text{A} \to \text{B C}, i, k, j) = \frac{r(\text{A} \to \text{B C}, i, k, j)}{\text{P}_{\text{IN}}(root, 0, n)}$ $\quad T_G = \text{argmax}_T \sum_{e \in T} q(e)$

MAX-RULE-PRODUCT: $q(\text{A} \to \text{B C}, i, k, j) = \frac{r(\text{A} \to \text{B C}, i, k, j)}{\text{P}_{\text{IN}}(root, 0, n)}$ $\quad T_G = \text{argmax}_T \prod_{e \in T} q(e)$

Figure 3: Different objectives for parsing with posteriors, yielding comparable results. A, B, C are nonterminal symbols, x, y, z are latent annotations and i, j, k are between-word indices. Hence (A_x, i, j) denotes a constituent labeled with A_x spanning from i to j. Furthermore, we write $e = (\text{A} \to \text{B C}, i, j, k)$ for brevity.

form. Exact match (likelihood) has this property. In general, however, we can approximate the expectation with samples from $P(T|w, G)$. The method for sampling derivations of a PCFG is given in Finkel et al. (2006) and Johnson et al. (2007). It requires a single inside-outside computation per sentence and is then efficient per sample. Note that for split grammars, a posterior parse sample can be drawn by sampling a derivation and projecting away the substates.

Fig. 2 shows the results of the following experiment. We constructed 10-best lists from the full grammar G in Sec. 2 using the parser of Petrov et al. (2006). We then took the same grammar and extracted 500-sample lists using the method of Finkel et al. (2006). The minimum risk parse candidate was selected for various loss functions. As can be seen, in most cases, risk minimization reduces test-set loss of the relevant quantity. Exact match is problematic, however, because 500 samples is often too few to draw a match when a sentence has a very flat posterior, and so there are many all-way ties.[2] Since exact match permits a non-sampled calculation of the expected risk, we show this option as well, which is substantially superior. This experiment highlights that the correct procedure for exact match is to find the most probable parse.

An alternative approach to reranking candidate parses is to work with inference criteria which admit dynamic programming solutions. Fig. 3 shows three possible objective functions which use the easily obtained posterior marginals of the parse tree distribution. Interestingly, while they have fairly different decision theoretic motivations, their closed-form solutions are similar.

[2]5,000 samples do not improve the numbers appreciably.

One option is to maximize likelihood in an approximate distribution. Matsuzaki et al. (2005) present a VARIATIONAL approach, which approximates the true posterior over parses by a cruder, but tractable sentence-specific one. In this approximate distribution there is no derivation / parse distinction and one can therefore optimize exact match by selecting the most likely derivation.

Instead of approximating the tree distribution we can use an objective function that decomposes along parse posteriors. The labeled brackets algorithm of Goodman (1996) has such an objective function. In its original formulation this algorithm maximizes the number of expected correct nodes, but instead we can use it to maximize the number of correct rules (the MAX-RULE-SUM algorithm). A worrying issue with this method is that it is ill-defined for grammars which allow infinite unary chains: there will be no finite minimum risk tree under recall loss (you can always reduce the risk by adding one more cycle). We implement MAX-RULE-SUM in a CNF-like grammar family where above each binary split is exactly one unary (possibly a self-loop). With this limitation, unary chains are not a problem. As might be expected, this criterion improves bracket measures at the expense of exact match.

We found it optimal to use a third approach, in which rule posteriors are multiplied instead of added. This corresponds to choosing the tree with greatest chance of having all rules correct, under the (incorrect) assumption that the rules correctness are independent. This MAX-RULE-PRODUCT algorithm does not need special treatment of infinite unary chains because it is optimizing a product rather than a sum. While these three methods yield

Objective	P	R	F1	EX
BEST DERIVATION				
Viterbi Derivation	89.6	89.4	89.5	37.4
RERANKING				
Random	87.6	87.7	87.7	16.4
Precision (sampled)	**91.1**	88.1	89.6	21.4
Recall (sampled)	88.2	**91.3**	89.7	21.5
F1 (sampled)	90.2	89.3	**89.8**	**27.2**
Exact (sampled)	89.5	89.5	89.5	25.8
Exact (non-sampled)	90.8	90.8	90.8	**41.7**
Exact/F1 (oracle)	95.3	94.4	95.0	63.9
DYNAMIC PROGRAMMING				
VARIATIONAL	90.7	90.9	90.8	**41.4**
MAX-RULE-SUM	90.5	**91.3**	90.9	40.4
MAX-RULE-PRODUCT	**91.2**	91.1	**91.2**	**41.4**

Table 2: A 10-best list from our best G can be reordered as to maximize a given objective either using samples or, under some restricting assumptions, in closed form.

very similar results (see Fig. 2), the MAX-RULE-PRODUCT algorithm consistently outperformed the other two.

Overall, the closed-form options were superior to the reranking ones, except on exact match, where the gains from correctly calculating the risk outweigh the losses from the truncation of the candidate set.

5 Multilingual Parsing

Most research on parsing has focused on English and parsing performance on other languages is generally significantly lower.[3] Recently, there have been some attempts to adapt parsers developed for English to other languages (Levy and Manning, 2003; Cowan and Collins, 2005). Adapting lexicalized parsers to other languages in not a trivial task as it requires at least the specification of head rules, and has had limited success. Adapting unlexicalized parsers appears to be equally difficult: Levy and Manning (2003) adapt the unlexicalized parser of Klein and Manning (2003) to Chinese, but even after significant efforts on choosing category splits, only modest performance gains are reported.

In contrast, automatically learned grammars like the one of Matsuzaki et al. (2005) and Petrov et al. (2006) require a treebank for training but no additional human input. One has therefore reason to believe that their performance will generalize better across languages than the performance of parsers that have been hand tailored to English.

[3]Of course, cross-linguistic comparison of results is complicated by differences in corpus annotation schemes and sizes, and differences in linguistic characteristics.

	ENGLISH (Marcus et al., 1993)	GERMAN (Skut et al., 1997)	CHINESE (Xue et al., 2002)
TrainSet	Section 2-21	Sentences 1-18,602	Articles 26-270
DevSet	Section 22	18,603-19,602	Articles 1-25
TestSet	Section 23	19,603-20,602	Articles 271-300

Table 3: Experimental setup.

5.1 Experiments

We trained models for English, Chinese and German using the standard corpora and splits as shown in Tab. 3. We applied our model directly to each of the treebanks, without any language dependent modifications. Specifically, the same model hyperparameters (merging percentage and smoothing factor) were used in all experiments.

Tab. 4 shows that automatically inducing latent structure is a technique that generalizes well across language boundaries and results in state of the art performance for Chinese and German. On English, the parser is outperformed only by the reranking parser of Charniak and Johnson (2005), which has access to a variety of features which cannot be captured by a generative model.

Space does not permit a thorough exposition of our analysis, but as in the case of English (Petrov et al., 2006), the learned subcategories exhibit interesting linguistic interpretations. In German, for example, the model learns subcategories for different cases and genders.

5.2 Corpus Variation

Related to cross language generalization is the generalization across domains for the same language. It is well known that a model trained on the Wall Street Journal loses significantly in performance when evaluated on the Brown Corpus (see Gildea (2001) for more details and the exact setup of their experiment, which we duplicated here). Recently McClosky et al. (2006) came to the conclusion that this performance drop is not due to overfitting the WSJ data. Fig. 4 shows the performance on the Brown corpus during hierarchical training. While the F_1 score on the WSJ is rising we observe a drop in performance after the 5th iteration, suggesting that some overfitting is occurring.

	≤ 40 words		all	
Parser	LP	LR	LP	LR
ENGLISH				
Charniak et al. (2005)	90.1	90.1	89.5	89.6
Petrov et al. (2006)	90.3	90.0	89.8	89.6
This Paper	**90.7**	**90.5**	**90.2**	**89.9**
ENGLISH (reranked)				
Charniak et al. (2005)[4]	**92.4**	**91.6**	**91.8**	**91.0**
GERMAN				
Dubey (2005)	F_1 76.3		-	
This Paper	**80.8**	**80.7**	**80.1**	**80.1**
CHINESE[5]				
Chiang et al. (2002)	**81.1**	78.8	78.0	75.2
This Paper	80.8	**80.7**	**78.8**	**78.5**

Table 4: Our final test set parsing performance compared to the best previous work on English, German and Chinese.

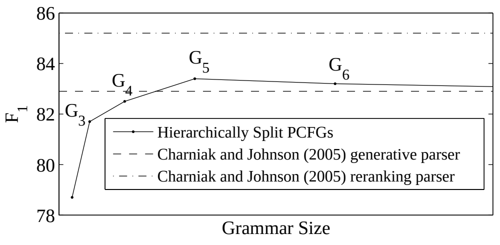


Figure 4: Parsing accuracy starts dropping after 5 training iterations on the Brown corpus, while it is improving on the WSJ, indicating overfitting.

6 Conclusions

The coarse-to-fine scheme presented here, in conjunction with the risk-appropriate parse selection methodology, allows fast, accurate parsing, in multiple languages and domains. For training, one needs only a raw context-free treebank and for decoding one needs only a final grammar, along with coarsening maps. The final parser is publicly available at `http://www.nlp.cs.berkeley.edu`.

Acknowledgments We would like to thank Eugene Charniak, Mark Johnson and Noah Smith for helpful discussions and comments.

References

E. Charniak and M. Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. In *ACL'05*.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. 6^{th} *Wkshop on Very Large Corpora*.

[4]This is the performance of the updated reranking parser available at http://www.cog.brown.edu/mj/software.htm

[5]Sun and Jurafsky (2004) report even better performance on this dataset but since they assume gold POS tags their work is not directly comparable (*p.c.*).

E. Charniak, M. Johnson, et al. 2006. Multi-level coarse-to-fine PCFG Parsing. In *HLT-NAACL '06*.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. In *Computational Linguistics*.

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

A. Corazza and G. Satta. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *HLT-NAACL '06*.

B. Cowan and M. Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *HLT-EMNLP '05*.

M. Dreyer and J. Eisner. 2006. Better informed training of latent syntactic features. In *EMNLP '06*, pages 317–326.

A. Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *ACL '05*.

J. Finkel, C. Manning, and A. Ng. 2006. Solving the problem of cascading errors: approximate Bayesian inference for lingusitic annotation pipelines. In *EMNLP '06*.

D. Gildea. 2001. Corpus variation and parser performance. *EMNLP '01*, pages 167–202.

J. Goodman. 1996. Parsing algorithms and metrics. *ACL '96*.

M. Johnson, T. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov Chain Monte Carlo. In *HLT-NAACL '07*.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03*, pages 423–430.

R. Levy and C. Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *ACL '03*, pages 439–446.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL '05*, pages 75–82.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *COLING-ACL'06*.

M. Mohri and B. Roark. 2006. Probabilistic context-free grammar induction based on structural zeros. In *HLT-NAACL '06*.

M.-J. Nederhof. 2005. A general technique to train language models on language models. In *Computational Linguistics*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL '06*, pages 443–440.

K. Sima'an. 1992. Computatoinal complexity of probabilistic disambiguation. *Grammars*, 5:125–151.

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free word order languages. In *Conference on Applied Natural Language Processing*.

M. Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, Massachusetts.

H. Sun and D. Jurafsky. 2004. Shallow semantic parsing of Chinese. In *HLT-NAACL '04*, pages 249–256.

R. Sutton and A. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

I. Titov and J. Henderson. 2006. Loss minimization in parse reranking. In *EMNLP '06*, pages 560–567.

K. Vijay-Shanker and A. Joshi. 1985. Some computational properties of Tree Adjoining Grammars. In *ACL '85*.

N. Xue, F.-D. Chiou, and M. Palmer. 2002. Building a large scale annotated Chinese corpus. In *COLING '02*.

Approximate Factoring for A* Search

Aria Haghighi, John DeNero, Dan Klein
Computer Science Division
University of California Berkeley
{aria42, denero, klein}@cs.berkeley.edu

Abstract

We present a novel method for creating A* estimates for structured search problems. In our approach, we project a complex model onto multiple simpler models for which exact inference is efficient. We use an optimization framework to estimate parameters for these projections in a way which bounds the true costs. Similar to Klein and Manning (2003), we then combine completion estimates from the simpler models to guide search in the original complex model. We apply our approach to bitext parsing and lexicalized parsing, demonstrating its effectiveness in these domains.

1 Introduction

Inference tasks in NLP often involve searching for an optimal output from a large set of structured outputs. For many complex models, selecting the highest scoring output for a given observation is slow or even intractable. One general technique to increase efficiency while preserving optimality is A* search (Hart et al., 1968); however, successfully using A* search is challenging in practice. The design of admissible (or nearly admissible) heuristics which are both effective (close to actual completion costs) and also efficient to compute is a difficult, open problem in most domains. As a result, most work on search has focused on non-optimal methods, such as beam search or pruning based on approximate models (Collins, 1999), though in certain cases admissible heuristics are known (Och and Ney, 2000; Zhang and Gildea, 2006). For example, Klein and Manning (2003) show a class of projection-based A* estimates, but their application is limited to models which have a very restrictive kind of score decomposition. In this work, we broaden their projection-based technique to give A* estimates for models which do not factor in this restricted way.

Like Klein and Manning (2003), we focus on search problems where there are multiple projections or "views" of the structure, for example lexical parsing, in which trees can be projected onto either their CFG backbone or their lexical attachments. We use general optimization techniques (Boyd and Vandenberghe, 2005) to approximately factor a model over these projections. Solutions to the projected problems yield heuristics for the original model. This approach is flexible, providing either admissible or nearly admissible heuristics, depending on the details of the optimization problem solved. Furthermore, our approach allows a modeler explicit control over the trade-off between the tightness of a heuristic and its degree of inadmissibility (if any). We describe our technique in general and then apply it to two concrete NLP search tasks: bitext parsing and lexicalized monolingual parsing.

2 General Approach

Many inference problems in NLP can be solved with agenda-based methods, in which we incrementally build hypotheses for larger items by combining smaller ones with some local configurational structure. We can formalize such tasks as graph search problems, where states encapsulate partial hypotheses and edges combine or extend them locally.[1] For example, in HMM decoding, the states are anchored labels, e.g. VBD$[5]$, and edges correspond to hidden transitions, e.g. VBD$[5] \rightarrow$ DT$[6]$.

The search problem is to find a minimal cost path from the start state to a goal state, where the path cost is the sum of the costs of the edges in the path.

[1]In most complex tasks, we will in fact have a hypergraph, but the extension is trivial and not worth the added notation.

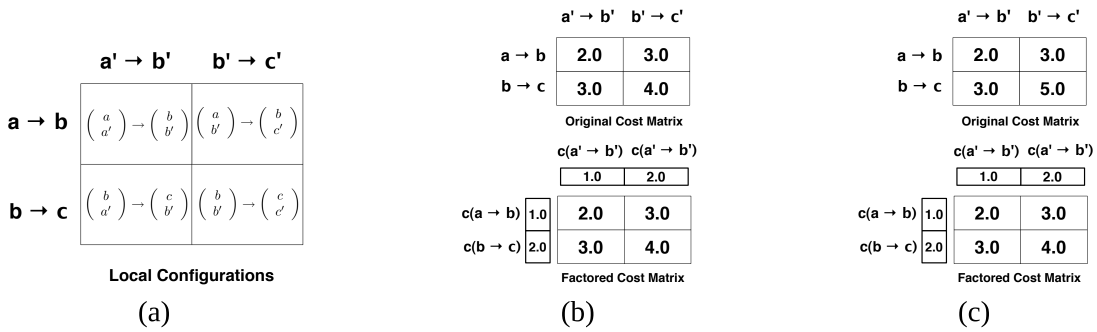


Figure 1: Example cost factoring: In (a), each cell of the matrix is a local configuration composed of two projections (the row and column of the cell). In (b), the top matrix is an example cost matrix, which specifies the cost of each local configuration. The bottom matrix represents our factored estimates, where each entry is the sum of configuration projections. For this example, the actual cost matrix can be decomposed exactly into two projections. In (c), the top cost matrix cannot be exactly decomposed along two dimensions. Our factored cost matrix has the property that each factored cost estimate is below the actual configuration cost. Although our factorization is no longer tight, it still can be used to produce an admissible heuristic.

For probabilistic inference problems, the cost of an edge is typically a negative log probability which depends only on some local configuration type. For instance, in PCFG parsing, the (hyper)edges reference anchored spans X$[i, j]$, but the edge costs depend only on the local rule type $X \rightarrow Y Z$. We will use a to refer to a local configuration and use $c(a)$ to refer to its cost. Because edge costs are sensitive only to local configurations, the cost of a path is $\sum_a c(a)$. A* search requires a *heuristic function*, which is an estimate $h(s)$ of the *completion cost*, the cost of a best path from state s to a goal.

In this work, following Klein and Manning (2003), we consider problems with *projections* or "views," which define mappings to simpler state and configuration spaces. For instance, suppose that we are using an HMM to jointly model part-of-speech (POS) and named-entity-recognition (NER) tagging. There might be one projection onto the NER component and another onto the POS component. Formally, a projection π is a mapping from states to some coarser domain. A state projection induces projections of edges and of the entire graph $\pi(G)$.

We are particularly interested in search problems with multiple projections $\{\pi_1, \ldots, \pi_\ell\}$ where each projection, π_i, has the following properties: its state projections induce well-defined projections of the local configurations $\pi_i(a)$ used for scoring, *and* the projected search problem admits a simpler inference. For instance, the POS projection in our NER-POS HMM is a simpler HMM, though the gains from this method are greater when inference in the projections have lower asymptotic complexity than the original problem (see sections 3 and 4).

In defining projections, we have not yet dealt with the projected scoring function. Suppose that the cost of local configurations decomposes along projections as well. In this case,

$$c\,(a) = \sum_{i=1}^{\ell} c_i(a)\ , \forall a \in \mathcal{A} \tag{1}$$

where $\mathcal{A}$ is the set of local configurations and $c_i(a)$ represents the cost of configuration a under projection π_i. A toy example of such a cost decomposition in the context of a Markov process over two-part states is shown in figure 1(b), where the costs of the joint transitions equal the sum of costs of their projections. Under the strong assumption of equation (1), Klein and Manning (2003) give an admissible A* bound. They note that the cost of a path decomposes as a sum of projected path costs. Hence, the following is an admissible additive heuristic (Felner et al., 2004),

$$h(s) = \sum_{i=1}^{\ell} h_i^*(s) \tag{2}$$

where $h_i^*(s)$ denote the optimal completion costs in the projected search graph $\pi_i(G)$. That is, the completion cost of a state bounds the sum of the completion costs in each projection.

In virtually all cases, however, configuration costs will not decompose over projections, nor would we expect them to. For instance, in our joint POS-NER task, this assumption requires that the POS and NER transitions and observations be generated independently. This independence assumption undermines the motivation for assuming a joint model. In the central contribution of this work, we exploit the projection structure of our search problem without making any assumption about cost decomposition.

Rather than assuming decomposition, we propose to find scores ϕ for the projected configurations which are *pointwise admissible*:

$$\sum_{i=1}^{\ell} \phi_i(a) \leq c(a), \forall a \in \mathcal{A} \tag{3}$$

Here, $\phi_i(a)$ represents a factored projection cost of $\pi_i(a)$, the π_i projection of configuration a. Given pointwise admissible ϕ_i's we can again apply the heuristic recipe of equation (2). An example of factored projection costs are shown in figure 1(c), where no exact decomposition exists, but a pointwise admissible lower bound is easy to find.

Claim. *If a set of factored projection costs $\{\phi_1, \ldots, \phi_\ell\}$ satisfy pointwise admissibility, then the heuristic from (2) is an admissible A* heuristic.*

Proof. Assume $a_1, \ldots, a_k$ are configurations used to optimally reach the goal from state s. Then,

$$\begin{aligned} h^*(s) &= \sum_{j=1}^{k} c(a_j) \geq \sum_{j=1}^{k}\sum_{i=1}^{\ell} \phi_i(a_j) \\ &= \sum_{i=1}^{\ell}\left(\sum_{j=1}^{k} \phi_i(a_j)\right) \geq \sum_{i=1}^{\ell} h_i^*(s) = h(s) \end{aligned}$$

□

The first inequality follows from pointwise admissibility. The second inequality follows because each inner sum is a completion cost for projected problem π_i and therefore $h_i^*(s)$ lower bounds it. Intuitively, we can see two sources of slack in such projection heuristics. First, there may be slack in the pointwise admissible scores. Second, the best paths in the projections will be overly optimistic because they have been decoupled (see figure 5 for an example of decoupled best paths in projections).

2.1 Finding Factored Projections for Non-Factored Costs

We can find factored costs $\phi_i(a)$ which are pointwise admissible by solving an optimization problem. We think of our unknown factored costs as a block vector $\phi = [\phi_1, .., \phi_\ell]$, where vector ϕ_i is composed of the factored costs, $\phi_i(a)$, for each configuration $a \in \mathcal{A}$. We can then find admissible factored costs by solving the following optimization problem,

$$\begin{aligned} &\underset{\phi}{\text{minimize}} \;\; \|\gamma\| \\ &\text{such that, } \gamma_a = c(a) - \sum_{i=1}^{\ell} \phi_i(a), \forall a \in \mathcal{A} \\ &\gamma_a \geq 0, \forall a \in \mathcal{A} \end{aligned} \tag{4}$$

We can think of each γ_a as the amount by which the cost of configuration a exceeds the factored projection estimates (the pointwise A* gap). Requiring $\gamma_a \geq 0$ insures pointwise admissibility. Minimizing the norm of the γ_a variables encourages tighter bounds; indeed if $\|\gamma\| = 0$, the solution corresponds to an exact factoring of the search problem. In the case where we minimize the 1-norm or ∞-norm, the problem above reduces to a linear program, which can be solved efficiently for a large number of variables and constraints.[2]

Viewing our procedure decision-theoretically, by minimizing the norm of the pointwise gaps we are effectively choosing a loss function which decomposes along configuration types and takes the form of the norm (i.e. linear or squared losses). A complete investigation of the alternatives is beyond the scope of this work, but it is worth pointing out that in the end we will care only about the gap on entire structures, not configurations, and individual configuration factored costs need not even be pointwise admissible for the overall heuristic to be admissible.

Notice that the number of constraints is $|\mathcal{A}|$, the number of possible local configurations. For many search problems, enumerating the possible configurations is not feasible, and therefore neither is solving an optimization problem with all of these constraints. We deal with this situation in applying our technique to lexicalized parsing models (section 4).

Sometimes, we might be willing to trade search optimality for efficiency. In our approach, we can explicitly make this trade-off by designing an alternative optimization problem which allows for slack

[2]We used the MOSEK package (Andersen and Andersen, 2000).

in the admissibility constraints. We solve the following soft version of problem (4):

$$\underset{\phi}{\text{minimize}} \;\; \|\gamma^{+}\| + C\|\gamma^{-}\| \qquad (5)$$

$$\text{such that, } \; \gamma_a = c(a) - \sum_{i=1}^{\ell} \phi_i(a), \forall a \in \mathcal{A}$$

where $\gamma^{+} = \max\{0, \gamma\}$ and $\gamma^{-} = \max\{0, -\gamma\}$ represent the componentwise positive and negative elements of γ respectively. Each $\gamma_a^{-} > 0$ represents a configuration where our factored projection estimate is not pointwise admissible. Since this situation may result in our heuristic becoming inadmissible if used in the projected completion costs, we more heavily penalize overestimating the cost by the constant C.

2.2 Bounding Search Error

In the case where we allow pointwise inadmissibility, i.e. variables γ_a^{-}, we can bound our search error. Suppose $\gamma_{\max}^{-} = \max_{a \in \mathcal{A}} \gamma_a^{-}$ and that L^* is the length of the longest optimal solution for the original problem. Then, $h(s) \leq h^*(s) + L^* \gamma_{\max}^{-}$, $\forall s \in \mathcal{S}$. This ϵ-admissible heuristic (Ghallab and Allard, 1982) bounds our search error by $L^* \gamma_{\max}^{-}$.[3]

3 Bitext Parsing

In bitext parsing, one jointly infers a synchronous phrase structure tree over a sentence w_s and its translation w_t (Melamed et al., 2004; Wu, 1997). Bitext parsing is a natural candidate task for our approximate factoring technique. A synchronous tree projects monolingual phrase structure trees onto each sentence. However, the costs assigned by a weighted synchronous grammar (WSG) $\mathcal{G}$ do not typically factor into independent monolingual WCFGs. We can, however, produce a useful surrogate: a pair of monolingual WCFGs with structures projected by $\mathcal{G}$ and weights that, when combined, underestimate the costs of $\mathcal{G}$.

Parsing optimally relative to a synchronous grammar using a dynamic program requires time $O(n^6)$ in the length of the sentence (Wu, 1997). This high degree of complexity makes exhaustive bitext parsing infeasible for all but the shortest sentences. In contrast, monolingual CFG parsing requires time $O(n^3)$ in the length of the sentence.

[3]This bound may be very loose if L is large.

3.1 A* Parsing

Alternatively, we can search for an optimal parse guided by a heuristic. The states in A* bitext parsing are rooted bispans, denoted $X\,[i,j] :: Y\,[k,l]$. States represent a joint parse over subspans $[i,j]$ of w_s and $[k,l]$ of w_t rooted by the nonterminals X and Y respectively.

Given a WSG $\mathcal{G}$, the algorithm prioritizes a state (or edge) e by the sum of its inside cost $\beta_{\mathcal{G}}(e)$ (the negative log of its inside probability) and its outside estimate $h(e)$, or completion cost.[4] We are guaranteed the optimal parse if our heuristic $h(e)$ is never greater than $\alpha_{\mathcal{G}}(e)$, the true outside cost of e.

We now consider a heuristic combining the completion costs of the monolingual projections of $\mathcal{G}$, and guarantee admissibility by enforcing point-wise admissibility. Each state $e = X\,[i,j] :: Y\,[k,l]$ projects a pair of monolingual rooted spans. The heuristic we propose sums independent outside costs of these spans in each monolingual projection.

$$h(e) = \alpha_s(X\,[i,j]) + \alpha_t(Y\,[k,l])$$

These monolingual outside scores are computed relative to a pair of monolingual WCFG grammars $\mathcal{G}_s$ and $\mathcal{G}_t$ given by splitting each synchronous rule

$$r = \begin{pmatrix} \mathrm{X}^{(s)} \\ \mathrm{Y}^{(t)} \end{pmatrix} \rightarrow \begin{pmatrix} \alpha\,\beta \\ \gamma\,\delta \end{pmatrix}$$

into its components $\pi_s(r) = X \rightarrow \alpha\beta$ and $\pi_t(r) = Y \rightarrow \gamma\delta$ and weighting them via optimized $\phi_s(r)$ and $\phi_t(r)$, respectively.[5]

To learn pointwise admissible costs for the monolingual grammars, we formulate the following optimization problem:[6]

$$\begin{aligned} &\underset{\gamma,\phi_s,\phi_t}{\text{minimize}} \;\; \|\gamma\|_1 \\ &\text{such that, } \gamma_r = c(r) - [\phi_s(r) + \phi_t(r)] \\ &\qquad\qquad \text{for all synchronous rules } r \in \mathcal{G} \\ &\qquad\qquad \phi_s \geq 0, \phi_t \geq 0, \gamma \geq 0 \end{aligned}$$

[4]All inside and outside costs are Viterbi, not summed.

[5]Note that we need only parse each sentence (monolingually) once to compute the outside probabilities for every span.

[6]The stated objective is merely one reasonable choice among many possibilities which require pointwise admissibility and encourage tight estimates.

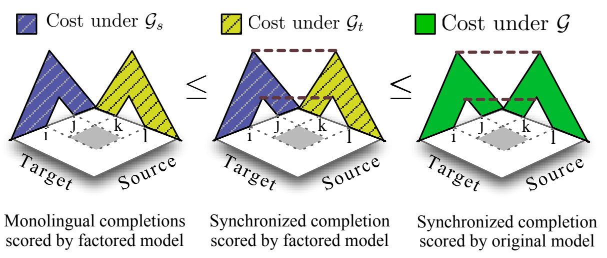

Cost under $\mathcal{G}_s$
Cost under $\mathcal{G}_t$
Cost under $\mathcal{G}$
$\leq$
$\leq$
i j k l
Target
Source
Monolingual completions scored by factored model
Synchronized completion scored by factored model
Synchronized completion scored by original model

Figure 2: The gap between the heuristic (left) and true completion cost (right) comes from relaxing the synchronized problem to independent subproblems and slack in the factored models.

(a) $$\begin{pmatrix} \text{NP}^{(s)} \\ \text{NP}^{(t)} \end{pmatrix} \rightarrow \begin{pmatrix} \text{NN}_1^{(s)}\ \text{NNS}_2^{(s)} \\ \text{NNS}_2^{(t)}\ \textit{de}\ \text{NN}_1^{(t)} \end{pmatrix}$$

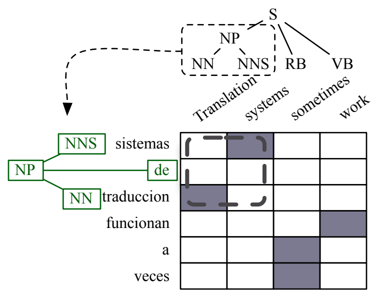
(b) 

Figure 3: (a) A tree-to-tree transducer rule. (b) An example training sentence pair that yields rule (a).

Figure 2 diagrams the two bounds that enforce the admissibility of $h(e)$. For any outside cost $\alpha_{\mathcal{G}}(e)$, there is a corresponding optimal completion structure o under $\mathcal{G}$, which is an outer shell of a synchronous tree. o projects monolingual completions o_s and o_t which have well-defined costs $c_s(o_s)$ and $c_t(o_t)$ under $\mathcal{G}_s$ and $\mathcal{G}_t$ respectively. Their sum $c_s(o_s) + c_t(o_t)$ will underestimate $\alpha_{\mathcal{G}}(e)$ by pointwise admissibility.

Furthermore, the heuristic we compute underestimates this sum. Recall that the monolingual outside score $\alpha_s(X\,[i,j])$ is the minimal costs for any completion of the edge. Hence, $\alpha_s(X\,[i,j]) \leq c_s(o_s)$ and $\alpha_t(X\,[k,l]) \leq c_t(o_t)$. Admissibility follows.

3.2 Experiments

We demonstrate our technique using the synchronous grammar formalism of tree-to-tree transducers (Knight and Graehl, 2004). In each weighted rule, an aligned pair of nonterminals generates two ordered lists of children. The non-terminals in each list must align one-to-one to the non-terminals in the other, while the terminals are placed freely on either side. Figure 3(a) shows an example rule.

Following Galley et al. (2004), we learn a grammar by projecting English syntax onto a foreign language via word-level alignments, as in figure 3(b).[7]

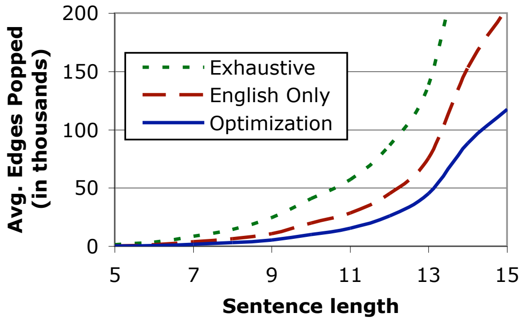
We parsed 1200 English-Spanish sentences using a grammar learned from 40,000 sentence pairs of the English-Spanish Europarl corpus.[8] Figure 4(a) shows that A* expands substantially fewer states while searching for the optimal parse with our *optimization* heuristic. The *exhaustive* curve shows edge expansions using the null heuristic. The intermediate result, labeled *English only*, used only the English monolingual outside score as a heuristic. Similar results using only Spanish demonstrate that both projections contribute to parsing efficiency. All three curves in figure 4 represent running times for finding the optimal parse.

Zhang and Gildea (2006) offer a different heuristic for A* parsing of ITG grammars that provides a forward estimate of the cost of aligning the unparsed *words* in both sentences. We cannot directly apply this technique to our grammar because tree-to-tree transducers only align non-terminals. Instead, we can augment our synchronous grammar model to include a lexical alignment component, then employ both heuristics. We learned the following two-stage generative model: a tree-to-tree transducer generates trees whose leaves are parts of speech. Then, the words of each sentence are generated, either jointly from aligned parts of speech or independently given a null alignment. The cost of a complete parse under this new model decomposes into the cost of the synchronous tree over parts of speech and the cost of generating the lexical items.

Given such a model, both our optimization heuristic and the lexical heuristic of Zhang and Gildea (2006) can be computed independently. Crucially, the sum of these heuristics is still admissible. Results appear in figure 4(b). Both heuristics (*lexical* and *optimization*) alone improve parsing performance, but their sum *opt+lex* substantially improves upon either one.

[7]The bilingual corpus consists of translation pairs with fixed English parses and word alignments. Rules were scored by their relative frequencies.

[8]Rare words were replaced with their parts of speech to limit the memory consumption of the parser.

(a) 

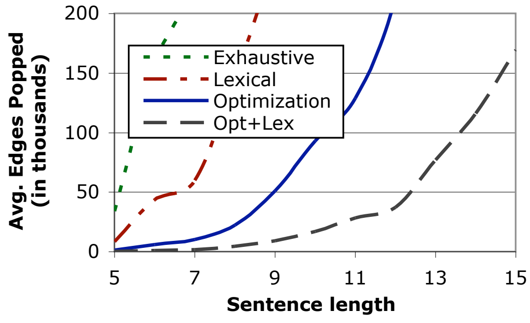
(b) 

Figure 4: (a) Parsing efficiency results with optimization heuristics show that both component projections constrain the problem. (b) Including a lexical model and corresponding heuristic further increases parsing efficiency.

4 Lexicalized Parsing

We next apply our technique to lexicalized parsing (Charniak, 1997; Collins, 1999). In lexicalized parsing, the local configurations are lexicalized rules of the form $X[h,t] \rightarrow Y[h',t']\ Z[h,t]$, where h, t, h', and t' are the head word, head tag, argument word, and argument tag, respectively. We will use $r = X \rightarrow YZ$ to refer to the CFG backbone of a lexicalized rule. As in Klein and Manning (2003), we view each lexicalized rule, ℓ, as having a CFG projection, $\pi_c(\ell) = r$, and a dependency projection, $\pi_d(\ell) = (h,t,h',t')$(see figure 5).[9] Broadly, the CFG projection encodes constituency structure, while the dependency projection encodes lexical selection, and both projections are asymptotically more efficient than the original problem. Klein and Manning (2003) present a factored model where the CFG and dependency projections are generated independently (though with compatible bracketing):

$$P(Y[h,t]Z[h',t'] \mid X[h,t]) = P(YZ|X)P(h',t'|t,h) \quad (6)$$

In this work, we explore the following non-factored model, which allows correlations between the CFG and dependency projections:

$$P(Y[h,t]Z[h',t'] \mid X[h,t]) = P(YZ|X,t,h) \; P(t'|t,Z,h',h)\ P(h'|t',t,Z,h',h) \quad (7)$$

This model is broadly representative of the successful lexicalized models of Charniak (1997) and Collins (1999), though simpler.[10]

4.1 Choosing Constraints and Handling Unseen Dependencies

Ideally we would like to be able to solve the optimization problem in (4) for this task. Unfortunately, exhaustively listing all possible configurations (lexical rules) yields an impractical number of constraints. We therefore solve a relaxed problem in which we enforce the constraints for only a subset of the possible configurations, $\mathcal{A}' \subseteq \mathcal{A}$. Once we start dropping constraints, we can no longer guarantee pointwise admissibility, and therefore there is no reason not to also allow penalized violations of the constraints we do list, so we solve (5) instead.

To generate the set of enforced constraints, we first include all configurations observed in the gold training trees. We then sample novel configurations by choosing (X,h,t) from the training distribution and then using the model to generate the rest of the configuration. In our experiments, we ended up with 434,329 observed configurations, and sampled the same number of novel configurations. Our penalty multiplier C was 10.

Even if we supplement our training set with many sample configurations, we will still see new projected dependency configurations at test time. It is therefore necessary to generalize scores from training configurations to unseen ones. We enrich our procedure by expressing the projected configuration costs as linear functions of features. Specifically, we define feature vectors $f_c(r)$ and $f_d(h,t,h't')$ over the CFG and dependency projections, and intro-

[9]We assume information about the distance and direction of the dependency is encoded in the dependency tuple, but we omit it from the notation for compactness.

[10]All probability distributions for the non-factored model are estimated by Witten-Bell smoothing (Witten and Bell, 1991) where conditioning lexical items are backed off first.

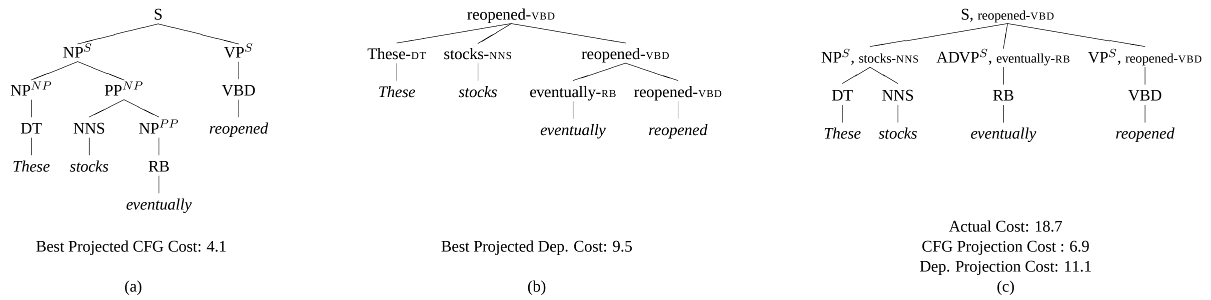


Figure 5: Lexicalized parsing projections. The figure in (a) is the optimal CFG projection solution and the figure in (b) is the optimal dependency projection solution. The tree in (c) is the optimal solution for the original problem. Note that the sum of the CFG and dependency projections is a lower bound (albeit a fairly tight one) on actual solution cost.

duce corresponding weight vectors w_c and w_d. The weight vectors are learned by solving the following optimization problem:

$$\begin{aligned} &\underset{\gamma, w_c, w_d}{\text{minimize}} \;\; \|\gamma^+\|^2 + C\|\gamma^-\|^2 \qquad (8) \\ &\quad \text{such that,} \;\; w_c \geq 0, w_d \geq 0 \\ &\qquad \gamma_\ell = c(\ell) - [w_c^T f_c(r) + w_d^T f_d(h,t,h',t')] \\ &\qquad \text{for } \ell = (r,h,t,h',t') \in \mathcal{A}' \end{aligned}$$

Our CFG feature vector has only indicator features for the specific rule. However, our dependency feature vector consists of an indicator feature of the tuple (h, t, h', t') (including direction), an indicator of the part-of-speech type (t, t') (also including direction), as well as a bias feature.

4.2 Experimental Results

We tested our approximate projection heuristic on two lexicalized parsing models. The first is the factored model of Klein and Manning (2003), given by equation (6), and the second is the non-factored model described in equation (7). Both models use the same parent-annotated head-binarized CFG backbone and a basic dependency projection which models direction, but not distance or valence.[11]

In each case, we compared A* using our approximate projection heuristics to exhaustive search. We measure efficiency in terms of the number of expanded hypotheses (edges popped); see figure 6.[12] In both settings, the factored A* approach substantially outperforms exhaustive search. For the factored model of Klein and Manning (2003), we can also compare our reconstructed bound to the known tight bound which would result from solving the pointwise admissible problem in (4) with all constraints. As figure 6 shows, the exact factored heuristic does outperform our approximate factored heuristic, primarily because of many looser, backed-off cost estimates for unseen dependency tuples. For the non-factored model, we compared our approximate factored heuristic to one which only bounds the CFG projection as suggested by Klein and Manning (2003). They suggest,

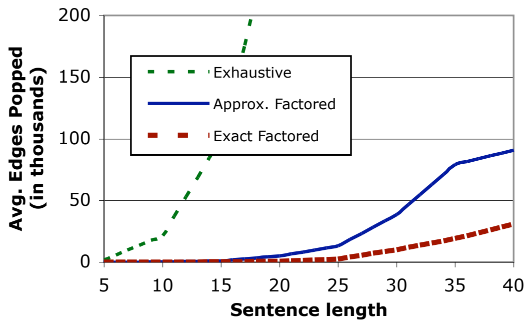
$$\phi_c(r) = \min_{\ell \in \mathcal{A}: \pi_c(\ell) = r} c(\ell)$$

where we obtain factored CFG costs by minimizing over dependency projections. As figure 6 illustrates, this CFG only heuristic is substantially less efficient than our heuristic which bounds both projections.

Since our heuristic is no longer guaranteed to be admissible, we evaluated its effect on search in several ways. The first is to check for search errors, where the model-optimal parse is not found. In the case of the factored model, we can find the optimal parse using the exact factored heuristic and compare it to the parse found by our learned heuristic. In our test set, the approximate projection heuristic failed to return the model optimal parse in less than 1% of sentences. Of these search errors, none of the costs were more than 0.1% greater than the model optimal cost in negative log-likelihood. For the non-factored model, the model optimal parse is known only for shorter sentences which can be parsed exhaustively. For these sentences up to length 15, there were no search errors. We can also check for violations of pointwise admissibility for configurations encoun-

[11]The CFG and dependency projections correspond to the PCFG-PA and DEP-BASIC settings in Klein and Manning (2003).

[12]All models are trained on section 2 through 21 of the English Penn treebank, and tested on section 23.

(a) 

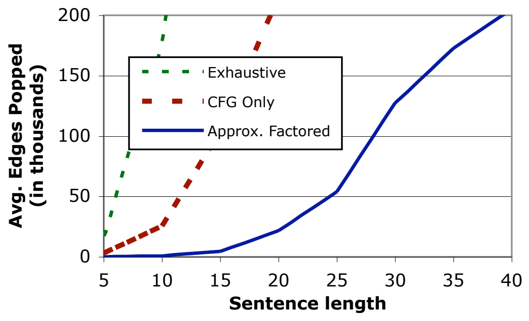
(b) 

Figure 6: Edges popped by exhaustive versus factored A* search. The chart in (a) is using the factored lexicalized model from Klein and Manning (2003). The chart in (b) is using the non-factored lexicalized model described in section 4.

tered during search. For both the factored and non-factored model, less than 2% of the configurations scored by the approximate projection heuristic during search violated pointwise admissibility.

While this is a paper about inference, we also measured the accuracy in the standard way, on sentences of length up to 40, using EVALB. The factored model with the approximate projection heuristic achieves an F_1 of 82.2, matching the performance with the exact factored heuristic, though slower. The non-factored model, using the approximate projection heuristic, achieves an F_1 of 83.8 on the test set, which is slightly better than the factored model.[13] We note that the CFG and dependency projections are as similar as possible across models, so the increase in accuracy is likely due in part to the non-factored model's coupling of CFG and dependency projections.

5 Conclusion

We have presented a technique for creating A* estimates for inference in complex models. Our technique can be used to generate provably admissible estimates when all search transitions can be enumerated, and an effective heuristic even for problems where all transitions cannot be efficiently enumerated. In the future, we plan to investigate alternative objective functions and error-driven methods for learning heuristic bounds.

Acknowledgments We would like to thank the anonymous reviewers for their comments. This work is supported by a DHS fellowship to the first author and a Microsoft new faculty fellowship to the third author.

[13]Since we cannot exhaustively parse with this model, we cannot compare our F_1 to an exact search method.

References

E. D. Andersen and K. D. Andersen. 2000. The MOSEK interior point optimizer for linear programming. In H. Frenk *et al.*, editor, *High Performance Optimization*. Kluwer Academic Publishers.

Stephen Boyd and Lieven Vandenberghe. 2005. *Convex Optimization*. Cambridge University Press.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *National Conference on Artificial Intelligence*.

Michael Collins. 1999. Head-driven statistical models for natural language parsing.

Ariel Felner, Richard Korf, and Sarit Hanan. 2004. Additive pattern database heuristics. *JAIR*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL*.

Malik Ghallab and Dennis G. Allard. 1982. A^*_ϵ - an efficient near admissible heuristic search algorithm. In *IJCAI*.

P. Hart, N. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*. IEEE.

Dan Klein and Christopher D. Manning. 2003. Factored A* search for models over sequences and trees. In *IJCAI*.

Kevin Knight and Jonathan Graehl. 2004. Training tree transducers. In *HLT-NAACL*.

I. Dan Melamed, Giorgio Satta, and Ben Wellington. 2004. Generalized multitext grammars. In *ACL*.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *ACL*.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*

Hao Zhang and Daniel Gildea. 2006. Efficient search for inversion transduction grammar. In *EMNLP*.

A Cascaded Machine Learning Approach to Interpreting Temporal Expressions

David Ahn **Joris van Rantwijk** **Maarten de Rijke**
ISLA, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{ahn, rantwijk, mdr}@science.uva.nl

Abstract

A new architecture for identifying and interpreting temporal expressions is introduced, in which the large set of complex hand-crafted rules standard in systems for this task is replaced by a series of machine learned classifiers and a much smaller set of context-independent semantic composition rules. Experiments with the TERN 2004 data set demonstrate that overall system performance is comparable to the state-of-the-art, and that normalization performance is particularly good.

1 Introduction

In order to fully understand a piece of text, we must understand its temporal structure. The first step toward such an understanding is identifying explicit references to time. We focus on the task of automatically annotating temporal expressions (or *timexes*)—both identifying them in text and interpreting them to determine what times they refer to. Timex annotation is more than normalizing date expressions. First, time consists of more than calendar dates and clock times—it also includes points of finer and coarser granularity, durations, and sets of times. Second, the expressions that refer to time are not just full date and time expressions—they may be underspecified, ambiguous, and anaphoric.

Building a system for the full timex identification and interpretation task can be tedious, requiring a great deal of manual effort. The 2004 Temporal Expression Recognition and Normalization (TERN) evaluation[1] evaluated systems on two tasks: timex *recognition* (identification) alone and recognition and *normalization* (interpretation) together. All the full-task systems were rule-based systems; the top performing full-task system uses in excess of one thousand hand-crafted rules, which probe words and their contexts in order to both identify timexes and to assemble information necessary to interpret them (Negri and Marseglia, 2004). By contrast, machine learned systems dominated the recognition-only task and even achieved slightly better recognition scores than their rule-based counterparts.

We seek to demonstrate that a timex annotation system that performs both recognition and normalization need not be a tangle of rules that serve double duty for identification and interpretation and that mix up context-dependent and context-independent processing. We propose a novel architecture that clearly separates syntactic, semantic, and pragmatic processing and factors out context-dependent from context-independent processing. Factoring out context-dependent disambiguation into separate classification tasks introduces the opportunity for using machine learning, which supports our main goal: building a portable, trainable timex annotation system in which the role of hand-crafted rules is minimized. The system we present here (available from `http://ilps.science.uva.nl/Resources/timextag/`) achieves the goal of making use of only a small set of hand-crafted, context-independent rules to achieve state-of-the-art normalization performance.

In the following section, we define what a timex is. We give an overview of our system architecture in §3 and describe the components in §4–7. §8 provides an evaluation of our system on the full timex annotation task, and we conclude in §9.

[1] `http://timex2.mitre.org/tern.html`

2 What is a timex?

Temporal semantics receives a great deal of attention in the semantics literature (cf. (Mani et al., 2005)), but the focus is generally on verbal semantics (i.e., tense and aspect). In determining what a timex is and how one should be normalized, we simply follow the TIDES TIMEX2 standard for timex annotation (Ferro et al., 2004). According to this standard, timexes are phrases or words that refer to times, where times may be points or durations, or sets of points or durations. Points are more than just instanteous moments in time—a point may also be a time with some duration, as long as it spans a single unit of some temporal granularity. Whether a timex refers to a point or a duration is a question of perspective rather than of ontology. A point-referring timex such as *October 18, 2006* refers to an interval of one day as an atom at the granularity of a day. A duration-referring timex such as *the whole day* may refer to the same temporal interval, but it focuses on the durative nature of this interval.

In addition to specifying which phrases are timexes, the TIMEX2 standard also provides a set of attributes for normalizing these timexes. We focus on the VAL attribute, which takes values that are an extension of the ISO-8601 standard for representing time (ISO, 1997). TIMEX2 VAL attributes can take one of three basic types of values:

Points are expressed as a string matching the pattern `dddd-dd-ddTdd:dd:dd.d+`, where `d` indicates a digit. Such a string is to be interpreted as *year-month-date*T*hour*:*minute*:*seconds*, and may be truncated from the right, indicating points of coarser granularity. Any place may be filled with a placeholder `X`, which indicates an unknown or vague value, and there are also a handful of token values (character strings) for seasons and parts of the day which may substitute for months and times. There is also an alternate week-based format `dddd-Wdd-d`, interpreted as *year*-W*week number-day of the week*.

Durations are expressed as a string matching the pattern `Pd+u` or `PTd+u`, where `d+` indicates one or more digits and `u` indicates a unit token (such as `Y` for years). A placeholder `X` may be used instead of a number to indicate vagueness.

Vague points: `past_ref`, `present_ref`, `future_ref`.

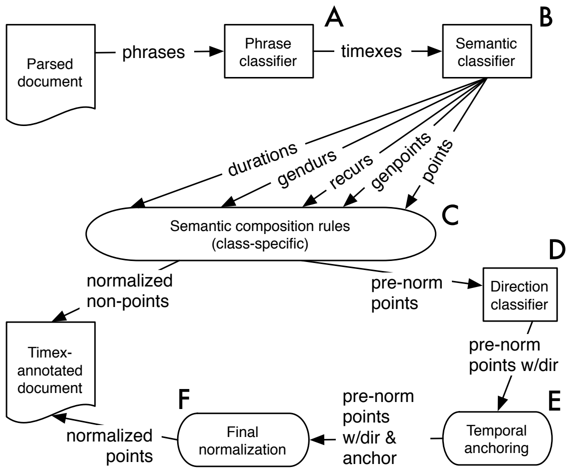


Figure 1: Timex annotation architecture (letters for ease of reference).

The other attribute which we address in this paper is the boolean-valued SET attribute; a SET timex is one that refers to a recurring time. The remaining attributes are MOD, ANCHOR_VAL, and ANCHOR_DIR; our system produces values for these attributes, but we do not address them in this paper.

The TIMEX2 annotation standard has been used to create several manually annotated corpora. For the experiments we present in this paper, we use the corpora annotated for the TERN 2004 evaluation (Ferro, 2004). These consist of a training set of 511 documents of newswire and broadcast news transcripts, with 5326 TIMEX2s, and a test set of 192 similar documents, with 1828 TIMEX2s.

3 Architecture

The architecture of our timex annotation system is depicted in Fig. 1. Our system begins with parsed documents as input. Our recognition module is a machine learned classifier (A); it is described in §4.

Phrases that have been classified as timexes are then sent to the semantic class classifier (B). Semantic class disambiguation is the first point at which context dependence enters into timex interpretation. While some timexes are unambiguous with respect to whether they refer to a point, a duration, or a set, many timexes are semantically ambiguous and can only be disambiguated in context. The machine learned classifier for this task is described in §5.

Based on the class assigned by the semantic class classifier, the semantic composition component (C) generates (underspecified) semantic representations using class-specific, context-independent rules. The rules we use are simple pattern-matching rules that map lexical items or sequences of lexical items within a timex to semantic representations. We describe the semantic composition component in §6.

For most classes of timexes, the semantic composition component generates a semantic representation that can be directly translated into a normalized value. Timexes that refer to specific points are the only exception. While some point timexes are fully qualified, and thus also directly normalizable, many need to be anchored to another time in context in order to be fully normalized. Thus, context dependence again enters the timex interpretation process, and now in two ways. One is obvious: these referential timexes, which need a temporal anchor, have to find it in context. This task requires a reference resolution process (E), which is described in §7.1.

The second ambiguity regards the relation between a referential timex and its anchor. Referential timexes, like anaphoric definites, relate to their anchors through a bridging relation, which is determined primarily by the content of the timex—e.g., *two years later* refers to a point two years after its anchor. For some referential timexes, though, the direction of the relation (before or after the anchor) is not specified. The machine learned classifier (D) resolves this ambiguity; see §7.2.

For referential timexes, final normalization (F) is a straightforward combination of semantic representation, temporal anchor, and direction class.

Not pictured in Fig. 1 is a module that recognizes and normalizes timexes in document metadata using a set of simple regular expressions (REs; 14 in total). This module also determines the document timestamp for referential timexes by using a few heuristics to choose from among multiple timestamps or a date from the document text, if necessary.

While our architecture is novel, we are not the first to modularize timex annotation systems. Even thoroughly rule-based systems (Negri and Marseglia, 2004; Saquete et al., 2002), separate temporal anchor tracking from the rest of the normalization process. The system of Mani and Wilson (2000) goes further in using separate sets of hand-crafted rules for recognition and normalization and in separating out several disambiguation tasks. Ahn et al. (2005b) decouple recognition from normalization—even using machine learning for recognition—and handle several disambiguation tasks separately. In none of these systems, though, are context-independent and context-dependent processing thoroughly separated, as here, and in all these systems, it is the rules that drive the processing—in both Mani et al. and Ahn et al.'s systems, sets of rules are used to determine which timexes need to be disambiguated.

4 Component A: Recognizing timexes

Systems that perform both recognition and normalization tend to take a rule-based approach to recognition (Mani and Wilson, 2000; Saquete et al., 2002; Schilder, 2004; Negri and Marseglia, 2004). Recognition-only systems are often based on machine learned classifiers (Hacioglu et al., 2005; Bethard and Martin, 2006), although some do use finite-state methods (Boguraev and Ando, 2005). Ahn et al. (2005a) find a benefit to decoupling recognition from normalization, and since our goal is to build a modular, trainable system, we take a machine-learning approach to recognition that is independent of our normalization components.

Generally, machine learned timex recognition systems reduce the task of identifying a timex *phrase* to one of classifying individual *words* by using (some variant of) B-I-O tagging, in which each word is tagged as (B)eginning, (I)nside, or (O)utside a timex phrase. Such a tagging scheme is not inherently sensitive to syntactic constituency and not well-suited to identifying nested timexes (but cf. (Hacioglu et al., 2005)). Considering that syntactic parsers are readily available, we have explored several ways of leveraging parse information in recognition, although we describe here only the method we use for experiments later in this paper.

We treat timex recognition as a binary *phrase* classification task: syntactic constituents are classified as timexes or non-timexes. We restrict classification to the following phrase types and lexical categories (based on (Ferro et al., 2004, §5)): NP, ADVP, ADJP, NN, NNP, JJ, CD, RB, and PP.[2] In order to identify candidate phrases and to extract

[2]We include PPs despite the TIDES guidelines, which explicitly exclude temporal PPs such as *before Thursday* because of prepositional modifiers such as *around* and *about*.

	Identification			Exact match		
	prec	rec	F	prec	rec	F
TEXT	0.912	0.786	0.844	0.850	0.732	0.787
DOC	0.929	0.813	0.867	0.878	0.769	0.819
BRO	0.973	0.891	0.930	0.905	0.829	0.865
BFT	0.976	0.880	0.926	0.885	0.798	0.839

Table 1: Recognition results: Identification.

parse-based features, we parse the TEXT elements of our documents with the Charniak parser (Charniak, 2000). Because of both parser and annotator errors, only 90.2% of the timexes in the training data align exactly with a parse, which gives an estimated upper-bound on recall using this method.

We use support vector machines for classification, in particular, the LIBSVM linear kernel implementation (Chang and Lin, 2001). The features we extract include character type patterns, lexical features such as weekday name and numeric year, a context window of two words to the left, and several parse-based features: the phrase type, the phrase head and initial word (and POS tag), and the dependency parent (and corresponding relation) of the head.

As with all our experiments in this paper, we train on the TERN training corpus and test on the test corpus. Our scores (precision, recall and F-measure for both identification (i.e., overlap) and exact-match) are given in Table 1, along with the scores of the best recognition-only (BRO) and full-task (BFT) TERN 2004 systems. Since our phrase classification method is only applied within document TEXT elements, we also present results using both our RE-based document metadata tagger and our phrase classifier for full documents (DOC). Only these scores can be compared with the TERN scores.

Our scores using this method approach those of the best systems, but there is still a gap, which, as we see in §8, affects our overall task performance.

5 Component B: Semantic classification

Timexes may refer to points, durations, or recurrences. While some timexes refer unambiguously to one of these, many timexes are ambiguous between two or even three of these (see (Hitzeman, 1993) for a theoretical semantic perspective on this ambiguity). Timexes may also refer generically or vaguely, which is another source of ambiguity.

While the TIMEX2 standard does not explicitly specify semantic classes in its annotations, the semantic classes we distinguish for our normalization system can be easily inferred from the form of the values of the attributes that are annotated, as follows:
Recurrence (recur): SET attribute set to true
Generic or vague duration (gendur): VAL begins with PX or PTX
Duration: VAL begins with P[0-9] or PT[0-9]
Generic or vague point (genpoint): Three possibilities: time-of-day w/o associated date expression (VAL begins with T[0-9]); general reference to past, present, or future (VAL is one of the vague tokens); date expression with unspecified high-order position (i.e., millennium position is X)
Point: Date expression with specified high-order position (may be precise or not—i.e., may include X at other positions—also may be of any granularity, from millennium down to hundredths of a second).

Resolving semantic class ambiguities is a context-dependent task that can be easily factored out of semantic interpretation, reducing the burden on the semantic interpretation rules. The classification task is straightforward: each timex must be classified into one of the five classes described above or into the null class (for timexes that have no VAL). Since the TERN data is not explicitly annotated for semantic class, we use the class definitions above to derive the semantic class of a timex from its VAL attribute.

We again use the LIBSVM linear kernel for classification, with the same features as for recognition. Even though some timexes are unambiguous with respect to semantic class, we train the classifier over all timexes, in the expectation that the contexts of unambiguous timexes will be similar enough to those of ambiguous timexes of the same class to help in classification. We compare the performance of our machine learned classifier to a heuristic baseline classifier that uses the head of the timex and the presence of numbers, names, and certain modifiers within the timex to decide how to classify it.

Table 2 gives the error rates, per class and overall, for the baseline and learned classifiers over phrase-aligned gold-standard timexes. The machine learned classifier halves the error rate of the baseline, mostly as a result of better performance on the duration and point classes. In §8, we see how this improvement in classification affects end-to-end performance.

Mani and Wilson (2000) and Ahn et al. (2005b)

classifier	overall	null	duration	gendur	genpoint	point	recur
BL	0.2085	1.0000	0.2534	0.0204	0.1462	0.1322	0.6087
SVM	0.1078	0.4143	0.1507	0.1020	0.1462	0.0496	0.2174
class dist	1290	70	146	49	253	726	46

Table 2: Error rates: semantic class.

class	rules	example
dur	13	Numeric -? (UNIT | UNITS)
gendur	3	(UNIT | UNITS)
genpt	21	(NUM24 | NUMWORD) o ' clock
point	31	ˆ Approx? DAYNAME? MONTHNAME .? Num31OrRank ,? YearNum
recur	11	(every | per) Numeric UNITS
misc	10	NUMWORD ((and | -)? NUMWORD)*

Table 3: Distribution of semantic composition rules.

also perform limited semantic class disambiguation. Both use machine learned classifiers to distinguish specific and generic uses of *today*, and Ahn et al. also use a machine learned classifier to disambiguate timexes between a point and a duration reading. Their error rate for this task is 27%, but since a set of heuristics is first used to select just ambiguous timexes, this score cannot be compared to ours.

6 Component C: Semantic composition

The semantic composition module uses context-independent, class-specific rules to compute for each timex an underspecified representation—a typed feature structure that depends on the timex's semantic class (features include unit and value for durations, year, month, date, and referential class for points; cf. (Dale and Mazur, 2006)). As the rules are not responsible for identification or class or direction disambiguation, they are fewer in number and simpler than in other systems (cf. 1000+ in (Negri and Marseglia, 2004)). Each rule consists of an RE-pattern, which may refer to a small lexicon of names, units, and numeric words, and is applied using a custom transducer. In total, there are 89 rules; Table 3 gives the distribution of rules and an example rule for each class. Tokens in `ALLCAPS` indicate lexical classes; tokens in `MixedCase` indicate other rules; and tokens in `lowercase` indicate lexical items.

7 Temporal anchors

Some point timexes are fully qualified, while others require a reference time, or temporal anchor, to be fully normalized.[3] There are three ways in which a temporal anchor is chosen for a timex. Some timexes, such as *today*, *three years ago*, and *next week*, are deictic and anchored to the time of speech (for us, the document timestamp). Others, such as *two months earlier* and *the next week*, are anaphoric and anchored to a salient time in discourse, just like an anaphoric pronoun or definite. The distinction between deictic and anaphoric timexes is not always clear-cut, since many anaphoric timexes, in the absence of an appropriate antecedent, are anchored deictically. A timex may also contain its own anchor: e.g., *two days after May 3*, whose anchor is the embedded anaphoric timex *May 3*.

Once a referential timex's temporal anchor has been determined, the value of the anchor must be combined with the timex, which may be either an offset or a name-like timex. Offsets, such as *two months earlier*, provide a unit u, a magnitude m, and optionally, a direction (before or after); the value of an offset is the point (of granularity u) that is m u units from its anchor in the indicated direction. Name-like timexes provide a position in a cycle, such as a day name within a week, and optionally, a direction. The value of a name-like timex is the time point bearing the name within the corresponding cycle of its anchor (or the immediately preceding or succeeding cycle, depending on the direction).

For both offsets and name-like timexes, the direction indication is optional. When no direction indication is given, the appropriate direction must be determined from context, as in this initial sentence from an article from 1998-11-28:

(1) A fundamentalist Muslim lawmaker has vowed to stop a shopping festival planned in *February*, a newspaper reported *Saturday*.

The first timex, *February*, clearly refers to the February following its anchor (the timestamp), while the second timex, *Saturday*, seems to refer to a point preceding its anchor (also the timestamp).

The next two sections describe our methods for temporal anchoring and direction classification.

[3]Our use of the term *temporal anchor* is distinct from the ANCHOR_VAL and ANCHOR_DIR attributes.

7.1 Component E: Temporal anchor tracking

Since temporal anchors are not annotated in the TIMEX2 standard, our system uses a simple heuristic method for temporal anchoring (cf. (Wiebe et al., 1997), who use a more complex rule-based system for timex anchoring in scheduling dialogues). Since we distinguish deictic and anaphoric timexes during semantic composition, we use a combination of two methods: for deictic timexes, the document timestamp is used, and for (some) anaphoric timexes, the most recent point timex, if it is fine-grained enough, is used as the temporal anchor (otherwise, the document timestamp is used). Because the documents in our corpora are short news texts, we actually treat anaphoric name-like points as deictic and use the most recent timex only for anaphoric offsets.

7.2 Component D: Direction classification

The idea of separating direction classification from the remainder of the normalization task is not new. (Mani and Wilson, 2000) use a heuristic method for this task, while (Ahn et al., 2005b) use a machine learned classifier. In contrast to Ahn et al., who use a set of heuristics to identify ambiguous timexes and train and test only on those, we train our classifier on all point and genpoint timexes and apply it to all point timexes. Genpoint timexes and many point timexes are not ambiguous w.r.t. direction, but we expect that the contexts of unambiguous timexes will be similar enough to those of ambiguous timexes of the same class to help classification.

Direction class is not annotated as part of the TIMEX2 standard. Given a temporal anchor tracking method, though, it is possible to derive imperfect direction class information from the VAL attribute. We use our anchor tracking method to associate each point and genpoint timex with an anchor and then compare the VAL of the timex with that of its anchor to decide what its direction class should be.

We again use the LIBSVM linear kernel for classification. We add two sets of features to those used for recognition and semantic classification. The first is inspired by Mani et al., who rely on the tense of neighboring verbs to decide direction class. Since verb tense alone is inherently deictic, it is not sufficient to decide the direction, but we do add both the closest verb (w.r.t. dependency paths) and its POS tag (as well as any verbs directly related to this verb) as features. The second set of features compares day names, month names, and years to the document timestamp. The comparison determines whether, within a single cycle of the appropriate granularity (week for day-names and year for month-names), the point named by the timex would be before, after, or the same as the point referred to by the timestamp.

classifier	overall	after	before	same
BL	0.1749	0.4587	0.0802	0.1934
SVM	0.2245	0.4404	0.1578	0.2305
SVM_VERB	0.2094	0.3119	0.1631	0.2346
SVM_ALL	0.1185	0.2110	0.0989	0.1070
class dist	726	109	374	243

Table 4: Error rates: direction class.

We compare our learned classifier with a heuristic baseline classifier which first checks for the presence of a year or certain modifiers such as *ago* or *next* in the timex; if that fails, it computes the date features described above for each word in the timex and returns `same` if any word compares to the timestamp as same; if that fails, it uses the tense of the nearest verb; and finally, it defaults to `same`.

Table 4 shows the results of applying our classifiers to all phrase-aligned gold-standard point timexes. BL is the baseline; SVM, SVM_VERB, and SVM_ALL are the classifiers learned using our basic feature set, the basic feature set plus the verb features, and all the features, respectively. The learned classifier using all the features reduces the error rate of the baseline classifier by about a third. Note, though, that the learned classifiers without the date comparison features (SVM and SVM_VERB) perform substantially worse than even the baseline. One reason for this becomes clear from Table 5, which gives the error rates for the classifiers restricted to timexes consisting solely of a month or a day name. Unlike points in general, these timexes are all ambiguous with respect to direction and are, in fact, the primary motivation for both Mani et al. and Ahn et al. to consider direction classification as a separate task.

These results demonstrate that the date comparison feature is responsible for a substantial reduction in error rate (over 85% from SVM to SVM_ALL) and that for the `same` class, performance is perfect. This is largely due to the writing style of the documents, in which the current day is often referred to by name

classifier	overall	after	before	same
BL	0.1000	0.4348	0.1061	0.0000
SVM	0.3647	0.6087	0.3485	0.3086
SVM_VERB	0.3176	0.3478	0.3485	0.2840
SVM_ALL	0.0529	0.1304	0.0909	0.0000
class dist	170	23	66	81

Table 5: Error rates: direction month/day.

instead of as *today*, as in example (1).

Although both Mani et al. and Ahn et al. build direction classifiers, neither provide comparable results. Mani et al. do not evaluate their direction heuristics at all, and Ahn et al. train and test their machine learned classifier only on timexes determined to be ambiguous by their heuristics. In any case, their error rate is significantly higher, at 38%.

8 End-to-end performance

We now consider the performance of the entire system and the contributions of the components. First, though, we discuss our evaluation metrics.

8.1 Scoring

The official TERN scoring script computes precision and recall for VAL only with respect to correctly recognized TIMEX2s with a non-null VAL. While this may be useful in determining how far behind normalization is from recognition for a given system, it does not provide an accurate picture of end-to-end system performance, since the recall base does not include all possible timexes and the precision base does not include incorrectly recognized timexes.

The scoring script provides several raw counts that can be used to compute measures that are more indicative of end-to-end performance: `actTIMEX2` (# of actually recognized TIMEX2s); `corrTIMEX2` (# of correctly recognized TIMEX2s); `posVAL` (# of correctly recognized TIMEX2s with a non-null gold VAL); `corrVAL` (# of correctly recognized TIMEX2s with a non-null gold VAL for which the system assigns the correct VAL); and `spurVAL` (# of correctly recognized TIMEX2s with null gold VAL for which the system assigns a VAL). With these counts, we can define `corrNOVAL` (# of correctly recognized TIMEX2s with a null gold VAL for which the system assigns a null VAL), as `corrTIMEX2` − `posVAL` − `spurVAL`. We then define end-to-end precision (absP) and recall (absR) as (`corrVAL` + `corrNOVAL`)/`actTIMEX2` and (`corrVAL`+`corrNOVAL`)/`possTIMEX2`, respectively. Official precision and recall for VAL are computed as `corrVAL`/`actVAL` and `corrVAL`/`possVAL`.

8.2 Results

Our first set of results (Table 6(Top)), which are restricted to timexes in document TEXT elements, compares our system (LLL) to a version of our system (BL) that uses the baseline classifiers for semantic and direction class. It also presents a series of oracle results that demonstrate the effect of swapping in perfect classification for each of the learned classifiers. The oracle runs are labeled with a three-letter code in which the first letter ((P)erfect or (L)earned) refers to phrase classification; the second, to semantic classification; and the third, to direction classification. Note: perfect phrase classification is not the same as perfect recognition, since it excludes timexes that fail to align with parsed phrases.

Using the learned classifiers (LLL), which reduce error rates by about one-half for semantic class and one-third for direction class over the baseline classifiers, results in a five-point improvement in absolute F-measure over the baseline system (BL). We also see from runs LLP, LPL, and LPP that further improvement of these classifiers would substantially improve end-to-end performance. Finally, we see from runs PLL and PPP that recognition performance is a major limiting factor in our end-to-end scores.

In Table 6(Bottom), we present results over full documents, including metadata and text. LLL and PLL are the same as before; ITC-IRST is the system of (Negri and Marseglia, 2004), which achieved the highest official F-measure in the TERN 2004 evaluation. The results of our system (LLL) are comparable to those of ITC-irst: because we recognize fewer timexes, our official F-measure is higher (0.899 vs. 0.872) while our absolute F-measure is lower (0.769 vs. 0.806). We see from run PLL that our recognition module is largely to blame—with perfect phrase classification for recognition, our normalization modules produce substantially better results. With a system such as ITC-irst's, it is not possible to separate recognition performance from normalization performance, since there is a single rule base that jointly performs the two tasks—all normalizable timexes are presumably already recognized.

System	corrVAL	corrNOVAL	actTIMEX2	P	R	F	absP	absR	absF
BL	859	32	1245	0.813	0.787	0.800	0.716	0.624	0.667
LLL	931	33	1245	0.882	0.853	0.867	0.774	0.676	0.722
LLP	938	33	1245	0.912	0.859	0.885	0.780	0.680	0.727
LPL	951	39	1245	0.916	0.871	0.893	0.795	0.694	0.741
LPP	987	39	1245	0.951	0.904	0.927	0.824	0.719	0.768
PLL	1008	63	1287	0.886	0.828	0.856	0.832	0.751	0.789
PPP	1097	70	1287	0.966	0.901	0.932	0.907	0.818	0.860
LLL	1285	33	1601	0.910	0.887	0.899	0.823	0.721	0.769
PLL	1362	63	1643	0.912	0.866	0.888	0.867	0.780	0.821
ITC-IRST	1365	35	1648	0.875	0.870	0.872	0.850	0.766	0.806

Table 6: Performance on VAL. (Top): TEXT-only. (Bottom): full document.

9 Conclusion

We have described a novel architecture for a timex annotation system that eschews the complex set of hand-crafted rules that is a hallmark of other systems. Instead, we decouple recognition from normalization and factor out context-dependent semantic and pragmatic processing from context-independent semantic composition. Our architecture allows us to use machine learned classifiers to make context-dependent disambiguation decisions, which in turn allows us to use a small set of simple, context-independent rules for semantic composition. The normalization performance of this system is competitive with the state of the art and our overall performance is limited primarily by recognition performance. Improvement in semantic and direction classification will yield further improvements in overall performance. Our other plans for the future include experimenting with dependency relations for semantic composition instead of lexical patterns, evaluating our temporal anchor tracking method, and training the full system on other corpora and adapting it for other languages.

Acknowledgement This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 600.065.120, 612-13- 001, 612.000.106, 612.066.302, 612.069.006, 640.001.501, 640.002.501, and by the E.U. IST programme of the 6th FP for RTD under project MultiMATCH contract IST-033104.

References

D. Ahn, S. Fissaha Adafre, and M. de Rijke. 2005a. Extracting temporal information from open domain text: A comparative exploration. In R. van Zwol, editor, *Proc. DIR'05*.

D. Ahn, S. Fissaha Adafre, and M. de Rijke. 2005b. Recognizing and interpreting temporal expressions in open domain texts. In S. Artemov et al., editors, *We Will Show Them: Essays in Honour of Dov Gabbay, Vol 1*, pages 31–50.

S. Bethard and J.H. Martin. 2006. Identification of event mentions and their semantic class. In *Proc. EMNLP 2006*.

B. Boguraev and R. Kubota Ando. 2005. TimeML-compliant text analysis for temporal reasoning. In *Proc. IJCAI-05*.

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

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL 2000*, pages 132–139.

R. Dale and P. Mazur. 2006. Local semantics in the interpretation of temporal expressions. In *Proc. Workshop on Annotating and Reasoning about Time and Events*, pages 9–16.

L. Ferro, L. Gerber, I. Mani, and G. Wilson, 2004. *TIDES 2003 Std. for the Annotation of Temporal Expressions*. MITRE.

L. Ferro. 2004. Annotating the TERN corpus. `http://timex2.mitre.org/tern_2004/ferro2_TERN2004_annotation_sanitized.pdf`.

K. Hacioglu, Y. Chen, and B. Douglas. 2005. Automatic time expression labeling for English and Chinese text. In A.F. Gelbukh, editor, *CICLing*, volume 3406 of *Lecture Notes in Computer Science*, pages 548–559.

J. Hitzeman. 1993. *Temporal Adverbials and the Syntax-Semantics Interface*. Ph.D. thesis, University of Rochester.

ISO. 1997. ISO 8601: Information interchange – representation of dates and times.

I. Mani and G. Wilson. 2000. Robust temporal processing of news. In *Proc. ACL'2000*, pages 69–76.

I. Mani, J. Pustejovsky, and R. Gaizauskas, editors. 2005. *The Language of Time: A Reader*. Oxford University Press.

M. Negri and L. Marseglia. 2004. Recognition and normalization of time expressions: ITC-irst at TERN 2004. Technical report, ITC-irst, Trento.

E. Saquete, P. Martínez-Barco, and R. Muñoz. 2002. Recognizing and tagging temporal expressions in Spanish. In *Workshop on Annotation Standards for Temporal Information in Natural Language, LREC 2002*, pages 44–51.

F. Schilder. 2004. Extracting meaning from temporal nouns and temporal prepositions. *ACM TALIP*.

J. Wiebe, T. O'Hara, K. McKeever, and T. Öhrström Sandgren. 1997. An empirical approach to temporal reference resolution. In *Proceedings of EMNLP-97*, pages 174–186.

Building and Refining Rhetorical-Semantic Relation Models

Sasha Blair-Goldensohn and
Google, Inc.
76 Ninth Avenue
New York, NY
sasha@google.com

Kathleen R. McKeown† and **Owen C. Rambow**‡
† Department of Computer Science
‡ Center for Computational Learning Systems
Columbia University
{kathy,rambow}@cs.columbia.edu

Abstract

We report results of experiments which build and refine models of rhetorical-semantic relations such as Cause and Contrast. We adopt the approach of Marcu and Echihabi (2002), using a small set of patterns to build relation models, and extend their work by refining the training and classification process using parameter optimization, topic segmentation and syntactic parsing. Using human-annotated and automatically-extracted test sets, we find that each of these techniques results in improved relation classification accuracy.

1 Introduction

Relations such as Cause and Contrast, which we call rhetorical-semantic relations (RSRs), may be signaled in text by cue phrases like *because* or *however* which join clauses or sentences and explicitly express the relation of constituents which they connect (Example 1). In other cases the relation may be implicitly expressed (2).[1]

Example 1 *Because of the recent accounting scandals, there have been a spate of executive resignations.*

Example 2 *The administration was once again beset by scandal. After several key resignations ...*

In this paper, we examine the problem of detecting such relations when they are not explicitly signaled. We draw on and extend the work of Marcu and Echihabi (2002). Our baseline model directly implements Marcu and Echihabi's approach, optimizing a set of basic parameters such as smoothing weights, vocabulary size and stoplisting. We then focus on improving the quality of the automatically-mined training examples, using topic segmentation and syntactic heuristics to filter out training instances which may be wholly or partially invalid. We find that the parameter optimization and segmentation-based filtering techniques achieve significant improvements in classification performance.

2 Related Work

Rhetorical and discourse theory has a long tradition in computational linguistics (Moore and Wiemer-Hastings, 2003). While there are a number of different relation taxonomies (Hobbs, 1979; McKeown, 1985; Mann and Thompson, 1988; Martin, 1992; Knott and Sanders, 1998), many researchers have found that, despite small differences, these theories have wide agreement in terms of the core phenomena for which they account (Hovy and Maier, 1993; Moser and Moore, 1996).

Work on automatic detection of rhetorical and discourse relations falls into two categories. Marcu and Echihabi (2002) use a pattern-based approach in mining instances of RSRs such as Contrast and Elaboration from large, unannotated corpora. We discuss this work in detail in Section 3. Other work uses human-annotated corpora, such as the RST Bank (Carlson et al., 2001), used by Soricut and Marcu (2003), the GraphBank (Wolf and Gibson, 2005), used by Wellner et al. (2006), or *ad-hoc* annotations, used by (Girju, 2003; Baldridge and Lascarides, 2005). In the past year, the ini-

[1]The authors would like to thank the four anonymous reviewers for helpful comments. This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.
The first author performed most of the research reported in this paper while at Columbia University.

Proceedings of NAACL HLT 2007, pages 428–435,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

tial public release of the Penn Discourse TreeBank (PDTB) (Prasad et al., 2006) has significantly expanded the discourse-annotated corpora available to researchers, using a comprehensive scheme for both implicit and explicit relations.

Some work in RSR detection has enlisted syntactic analysis as a tool. Marcu and Echihabi (2002) filter training instances based on Part-of-Speech (POS) tags, and Soricut and Marcu (2003) use syntactic features to identify sentence-internal RST structure. Lapata and Lascarides (2004) focus their work syntactically, analyzing temporal links between main and subordinate clauses. Sporleder and Lascarides (2005) extend Marcu and Echihabi's approach with the addition of a number of features, including syntactic features based on POS and argument structure, as well as lexical and other surface features. They report that, when working with sparse training data, this richer feature set, combined with a boosting-based algorithm, achieves more accurate classification than Marcu and Echihabi's simpler, word-pair based approach (we describe the latter in the next section).

3 The M&E Framework

We model two RSRs, Cause and Contrast, adopting the definitions of Marcu and Echihabi (2002) (henceforth M&E) for their Cause-Explanation-Evidence and Contrast relations, respectively. In particular, we follow their intuition that in building an automated model it is best to adopt a higher-level view of relations (cf. (Hovy and Maier, 1993)), collapsing the finer-grained distinctions that hold within and across relation taxonomies.

M&E use a three-stage approach common in corpus linguistics: collect a large set of class instances (*instance mining*), analyze them to create a model of differentiating features (*model building*), and use this model as input to a *classification* step which determines the most probable class of unknown instances.

The intuition of the M&E model is to apply a set of RSR-associated cue phrase patterns over a large text corpus to compile a training set without the cost of human annotation. For instance, Example 1 will match the Cause-associated pattern "Because of W_1 , W_2 .", where W_1 and W_2 stand for non-empty strings containing word tokens. In the aggregate, such instances increase the prior belief that, e.g., a text span containing the word *scandals* and one containing *resignations* are in a Cause relation. A critical point is that the cue words themselves (e.g., *because*) are discarded before extracting these word pairs; otherwise these cue phrases themselves would likely be the most distinguishing features learned.

More formally, M&E build up their model through the three stages mentioned above as follows: In *instance mining*, for each RSR r they compile an instance set I_r of (W_1, W_2) spans which match a set of patterns associated with r. In *model building*, features are extracted from these instances; M&E extract a single feature, namely the frequency of token pairs derived from taking the cartesian product of $W_1 = \{w_1...w_n\} \times W_2 = \{w_{n+1}...w_m\} = \{(w_1, w_{n+1})...(w_n, w_m)\}$ over each span pair instance $(W_1, W_2) \in I$; these pair frequencies are tallied for each RSR into a frequency table F_r. Then in *classification*, the most likely relation r between two unknown-relation spans W_1 and W_2 can be determined by a naïve Bayesian classifier as $\mathrm{argmax}_{r \in R} P(r|W_1, W_2)$, where the probability $P(r|W_1, W_2)$ is simplified by assuming the independence of the individual token pairs to: $\prod_{(w_i,w_j) \in W_1,W_2} P((w_i, w_j)|r)$. The frequency counts F_r are used as maximum likelihood estimators of $P((w_i, w_j)|r)$.

4 TextRels

TextRels is our implementation of the M&E framework, and serves as our platform for the experiments which follow.

For *instance mining*, we use a set of cue phrase patterns derived from published lists (e.g., (Marcu, 1997; Prasad et al., 2006)) to mine the Gigaword corpus of 4.7 million newswire documents[2] for relation instances. We mine instances of the Cause and Contrast RSRs discussed earlier, as well as a NoRel "relation". NoRel is proposed by M&E as a default model of same-topic text across which no specific RSR holds; instances are extracted by taking text span pairs which are simply sentences from the same document separated by at least three intervening sentences. Table 1 lists a sample of our ex-

[2]distributed by the Linguistic Data Consortium

Type	**Sample Patterns**	**Instances**	**Instances, M&E**
Cause	*BOS* Because W_1 , W_2 *EOS* *BOS* W_1 *EOS BOS* Therefore , W_2 *EOS*.	926,654	889,946
Contrast	*BOS* W_1 , but W_2 *EOS* *BOS* W_1 *EOS BOS* However , W_2 *EOS*.	3,017,662	3,881,588
NoRel	*BOS* W_1 *EOS (BOS EOS){3,} BOS* W_2 *EOS*	1,887,740	1,000,000

Table 1: RSR types, sample extraction patterns, number of training instances used in TextRels, and number of training instances used by M&E. BOS and EOS are sentence beginning/end markers.

traction patterns and the total number of training instances per relation; in addition, we hold out 10,000 instances of each type, which we divide evenly into development and training sets.

For *model building*, we compile the training instances into token-pair frequencies. We implement several parameters which control the way these frequencies are computed; we discuss these parameters and their optimization in the next section.

For *classification*, we implement three binary classifiers (for Cause vs Contrast, Cause vs NoRel and Contrast vs NoRel) using the naïve Bayesian framework of the M&E approach. We implement several classification parameters, which we discuss in the next section.

5 Parameter Optimization

Our first set of experiments examine the impact of various parameter settings in TextRels, using classification accuracy on a development set as our heuristic. We find that the following parameters have strong impacts on classification:

• *Tokenizing* our training instances using stemming slightly improves accuracy and also reduces model size.

• *Laplace smoothing* is as accurate as Good-Turing, but is simpler to implement. Our experiments find peak performance with 0.25 λ value, i.e. the frequency assumed for unseen pairs.

• *Vocabulary size* of 6,400 achieves peak performance; tokens which are not in the most frequent 6,400 stems (computed over Gigaword) are replaced by an UNK pseudo-token before F is computed.

• *Stoplisting* has a negative impact on accuracy; we find that even the most frequent tokens contribute useful information to the model; a stoplist size of zero achieves peak performance.

• *Minimum Frequency* cutoff is imposed to discard from F token pair counts with a frequency of < 4; results degrade slightly below this value, and discarding this long tail of rare pair counts significantly shrinks model size.

Classif. / TestSet	Pdtb Opt	Seg	Auto Opt	Seg	Auto-S Opt	Seg	M&E
Cau/Con	59.1	61.1	69.8	69.7	70.3	70.6	87
Cau/NR	75.2	74.3	72.7	73.5	71.2	72.3	75
Con/NR	67.4	69.7	70.7	71.3	68.2	70.0	64

Table 2: Classifier accuracy across PDTB, Auto and Auto-S test sets for the parameter-optimized classifier ("Opt") and the same classifier trained on segment-constrained instances ("Seg"). Accuracy from M&E is reported for reference, but we note that they use a different test set so the comparison is not exact. Baseline in all cases is 50%.

To evaluate the performance of our three binary classifiers using these optimizations, we follow the protocol of M&E. We present the classifier for, e.g., Cause vs NoRel with an equal number of span-pair instances for each RSR (as in training, any pattern text has been removed). We then determine the accuracy of the classifier in predicting the actual RSR of each instance; in all cases we use an equal number of input pairs for each RSR so random baseline is 50 %. We carry out this evaluation over two different test sets.

The first set ("PDTB") is derived from the Penn Discourse TreeBank (Prasad et al., 2006). We extract "Implicit" relations, i.e. text spans from adjacent sentences between which annotators have inferred semantics not marked by any surface lexical item. To extract test instances for our Cause RSR, we take all PDTB Implicit relations marked with "Cause" or "Consequence" semantics (344 total instances); for our Contrast RSR, we take instances marked with "Contrast" semantics (293 total instances).[3] PDTB marks the two "Arguments" of these relationship instances, i.e. the text spans to which they apply; these are used as test (W_1, W_2) span pairs for classification. We test the performance on PDTB data using 280 randomly selected instances each from the PDTB Cause and Contrast sets, as well as 280 randomly selected instances from our test set of automatically extracted NoRel instances (while there is a NoRel relation included in PDTB, it is too sparse to use in this testing, with 53 total examples).

The second test set ("Auto") uses the 5,000 test instances of each RSR type automatically extracted in our instance mining process.

Table 2 lists the accuracy for the optimized ("Opt") classifier over the Auto and PDTB test sets[4]. (The "Seg" columns and "Auto-S" test set are explained in the next section.)

We also list for reference the accuracy reported by M&E; however, their training and test sets are not the same so this comparison is inexact, although their test set is extracted automatically in the same manner as ours. In the Cause versus Contrast case, their reported performance exceeds ours significantly; however, in a subset of their experiments which test Cause versus Contrast on instances from the human annotated RSTBank corpus (Carlson et al., 2001) where no cue phrase is present, they report only 63% accuracy over a 56% baseline (the baseline is $> 50\%$ because the number of input examples is unbalanced).

Since we also experience a drop in performance from the automatically derived test set to the human-annotated test set (the PDTB in our case), we further examined this issue. Our goal was to see if the lower accuracy on the PDTB examples is due to (1) the inherent difficulty of identifying implicit relation spans or (2) something else, such as the corpus-switching effect due to our model being trained and tested on different corpora (Gigaword and PDTB, respectively). To informally test this, we tested against explicitly cue-phrase marked examples gathered from PDTB. That is, we used the M&E-style method for mining instances, but we gathered them from the PDTB corpus. Interestingly, we found that (1) appears to be the case: for the Cause vs. Contrast (68.7%), Cause vs. NoRel (73.0%) and (Contrast vs. NoRel (71.0%) classifiers, the performance patterns with the Auto test set rather than the results from the PDTB Implicit test set. This bolsters the argument that "synthetic" implicit relations, i.e. those created by stripping of originally present cue phrases, cannot be treated as fully equivalent to "organic" ones annotated by a human judge but which are not explicitly indicated by a cue phrase. Sporleder and Lascarides (To Appear) recently investigated this issue in greater detail, and indeed found that such synthetic and organic instances appear to have important differences.

6 Using Topic Segmentation

In our experiments with topic segmentation, we augmented the instance mining process to take account of topic segment boundaries. The intuition here is that all sentence boundaries should not be treated equally during RSR instance mining. That is, we would like to make our patterns recognize that some sentence boundaries indicate merely an orthographic break without a switch in topic, while others can separate quite distinct topics. Sometimes the latter type are marked by paragraph boundaries, but these are unreliable markers since they may be used quite differently by different authors.

Instead, we take the approach of adding topic segment boundary markers to our corpus, which we can then integrate into our RSR extraction patterns. In the case of NoRel, our assumption in our original patterns is that the presence of at least three intervening sentences is a sufficient heuristic for finding spans which are not joined by one of the other RSRs; we add the constraint that sentences in a NoRel relation be in distinct topical segments, we can increase model quality. Conversely, for two-sentence Cause and Contrast instances, we add the constraint that there must *not* be an intervening topic segment boundary between the two sentences.

[3]Note that we are using the initial PDTB release, in which only three of 24 data sections have marked Implicit relations, so that the number of such examples will presumably grow in the next release.

[4]We do not provide pre-optimization baseline accuracy because this would be arbitrarily depend on how sub-optimally we select values select parameter values. For instance, by using a Vocabulary Size of 3,200 (rather than 6,400) and a Laplace λ value of 1, the mean accuracy of the classifiers on the Auto test set drops from 71.6 to 70.5; using a Stoplist size of 25 (rather than 0) drops this number to 67.3.

Before applying these segment-augmented patterns, we must add boundary markers to our corpus. While the concept of a topic segment can be defined at various granularities, we take a goal-oriented view and aim to identify segments with a mean length of approximately four sentences, reasoning that these will be long enough to exclude some candidate NoRel instances, yet short enough to exclude a non-trivial number of Contrasts and Cause instances. We use an automatic topic segmentation tool, LCSeg (Galley et al., 2003) setting parameters so that the derived segments are of the approximate desired length. Using these parameters, LCSeg produces topic segments with a mean length of 3.51 sentences over Gigaword, as opposed to 1.54 sentences for paragraph boundaries. Using a simple metric that assumes "correct" segment boundaries always occur at paragraph boundaries, LCSeg achieves 76% precision.

We rerun the instance mining step of TextRels over the segmented training corpus, after adding the segment-based constraints mentioned above to our pattern set. Although our constraints reduce the overall number of instances available in the corpus, we extract for training the same number of instances per RSR as listed in Table 1 (our non-segment-constrained training set does not use all instances in the corpus). Using the optimal parameter settings determined in the previous section, we build our models and classifiers based on these segment-constrained instances.

To evaluate the classifiers built on the segment-constrained instances, we can essentially follow the same protocol as in our Parameter Optimization experiments. However, we must choose whether to use a held-out test set taken from the segment-constrained instances ("Auto-S") or the same test set as used to evaluate our parameter optimization, i.e. the ("Auto") test set from unsegmented training data. We decide to test on both. On the one hand, segmentation is done automatically, so it is realistic that given a "real world" document, we can compute segment boundaries to help our classification judgments. On the other hand, testing on unsegmented input allows us to compare more directly to the numbers from our previous section. Further, for tasks which would apply RSR models outside of a single-document context (e.g., for assessing coherence of a synthesized abstract), a test on unsegmented input may be more relevant. Table 2 shows the results for the "Seg" classifiers on both Auto test sets, as well as the PDTB test set.

We observe that the performance of the classifiers is indeed impacted by training on the segment-constrained instances. On the PDTB test data, performance using the segment-trained classifiers improves in two of three cases, with a mean improvement of 1.2%. However, because of the small size of this set, this margin is not statistically significant.

On the automatically-extracted test data, the segment-trained classifier is the best performer in all three cases when using the segmented test data; while the margin is not statistically significant for a single classifier, the overall accurate-inaccurate improvement is significant ($p < .05$) using a Chi-squared test. On the unsegmented test data, the segment-trained classifiers are best in two of three cases, but the overall accurate-inaccurate improvement does not achieve statistical significance. We conclude tentatively that a classifier trained on examples gleaned with topic-segment-augmented patterns performs more accurately than our baseline classifier.

7 Using Syntax

Whether or not we use topic segmentation to constrain our training instances, our patterns rely on sentence boundaries and cue phrase anchors to demarcate the extents of the text spans which form our RSR instances. However, an instance which matches such a pattern often contains some amount of text which is not relevant to the relation in question. Consider:

Example 3 *Wall Street investors, citing* **a drop in oil prices because weakness in the automotive sector***, sold off shares in GM today.*

In this case, a syntactically informed analysis could be used to extract the constituents in the cause-effect relationship from within the boldfaced nominal clause only, i.e. as "a drop in oil prices" and "weakness in the automotive sector." However, the output of our instance mining process simply splits the string around the cue phrase "because of" and extracts the entire first and second parts of the sentence as the constituents. Of course, this may be for the best; in this case there is an implicit Cause relationship between the NP headed by *drop* and the *sold* VP which our pattern-based rules inadvertently capture; our experiments here test whether such noise is more helpful than hurtful.

Recognizing the potential complexity of using syntactic phenomena, we reduce the dimensions of the problem. First, we focus on single-sentence instances; this means we analyze only Cause and Contrast patterns, since NoRel uses only multi-sentence patterns. Second, within the Cause and Contrast instances, we narrow our investigation to the most productive pattern of each type (in terms of training instances extracted), given that different syntactic phenomena may be in play for different patterns. The two patterns we use are "W_1 because W_2" for Cause (accounts for 54% of training instances) and "W_1 , but W_2" for Contrast (accounts for 41% of training instances). Lastly, we limit the size of our training set because of parsing time demands. We use the Collins parser (Collins, 1996) to parse 400,000 instances each of Cause and Contrast for our final results. Compared with our other models, this is approximately 43% of our total Cause instances and 13% of our total Contrast instances. For the NoRel model, we use a randomly selected subset of 400,000 instances from our training set. For all relations, we use the non-segment-constrained instance set as the source of these instances.

7.1 Analyzing and Classifying Syntactic Errors

To analyze the possible syntactic bases for the type of over-capturing behavior shown in Example 3, we create a small development set of 100 examples each from Cause and Contrast training examples which fit the criteria just mentioned. We then manually identify and categorize any instances of over-capturing, labeling the relation-relevant and irrelevant spans. We find that 75% of Cause and 58% of Contrast examples contain at least some over-capturing; we observe several common reasons for over-capturing that we characterize syntactically. For example, a matrix clause with a verb of saying should not be part of the RSR. Using automatic parses of these instances created by we then design syntactic filtering heuristics based on a manual examination of parse trees of several examples from our development set.

For Contrast, we find that using the coordinating conjunction (CC) analysis of *but*, we can use a straightforward rule which limits the extent of RSR spans captured to the conjuncts/children of the CC node, e.g. by capturing only the boldfaced clauses in the following example:

Example 4 *For the past six months, management has been* **revamping positioning and strategy***, but* **also scaling back operations**.

This heuristic successfully cuts out the irrelevant temporal relative clause, retaining the relevant VPs which are being contrasted. Note that the heuristic is not perfect; ideally the adverb *also* would be filtered here, but this is more difficult to generalize since contentful adverbials, e.g. *strategically* should not be filtered out.

For the *because* pattern, we capture the right-hand span as any text in child(ren) nodes of the *because* IN node. We extend the left-hand span only as far as the first phrasal (e.g. VP) or finite clause (e.g. SBAR) node above the *because* node. Analyzing Example 3, the heuristic correctly captures the right-hand span; however, to the left of *because*, the heuristic cuts too much, and misses the key noun *drop*.

7.2 Error Analysis: Evaluating the Heuristics

The first question we ask is, how well do our heuristics work in identifying the actual correct RSR extents? We evaluate this against the Penn Discourse TreeBank (PDTB), restricting ourselves to discourse-annotated *but* and *because* sentences which match the RSR patterns which are the subject of our syntactic filtering. Since the PDTB is annotated on the same corpus as Penn TreeBank (PTB), we separately evaluate the performance of our heuristics using gold-standard PTB parses ("PDTB-Gold") versus the trees generated by Collins' parser ("PDTB-Prs"). We extract our test data from the PDTB data corresponding to section 23 of PTB, i.e. the standard testing section, so that the difference between the gold-standard and real parse trees is meaningful. Section 23 contains 60 annotated instances of *but* and 52 instances of *because* which we can use for this purpose. We define the measurement of accuracy here in terms of word-level precision/recall. That is, the set of words filtered by our heuristics are compared to the "correct"

Heuristic	PDTB-Prs	PDTB-Gold
Contrast	89.6 / 73.0 / 80.5	79.0 / 80.6 / 79.8
Cause	78.5 / 78.8 / 78.6	87.3 / 79.5 / 83.2

Table 3: Precision/Recall/F-measure of syntactic heuristics under various data sets and settings as described in Section 7.2.

words to cut, i.e. those which the annotated RSR extents exclude. The results of this analysis are shown in Table 3.

We performed an analysis of our heuristics on Section 24 of the PDTB. In that section, there are 74 relevant sentences: 20 sentences with *because*, and 54 sentences with *but*. Exactly half of all sentences (37) have no problems in the application of the heuristics (7 *because* sentences, 30 *but* sentences). Among the remaining sentences, the main source of problems is that our heuristics do not always remove matrix clauses with verbs of saying (15 cases total, 8 of which are *because* sentences). For the *but* clauses, our heuristics removed the subject in 12 cases where the PDTB did not do so. Additionally, the heuristic for *but* sentences does not correctly identify the second conjunct in five cases (choosing instead a parenthetical, for instance).

In looking at our syntactic heuristics for the Cause relationship, we see that they indeed eliminate the most frequent source of discrepancies with the PDTB, namely the false inclusion of a matrix clause of saying, resulting in 15 out of 20 perfect analyses.

We also evaluate the difference in performance between the PDTB-Gold and PDTB-Prs performance to determine to what extent using a parser (as opposed to the Gold Standard) degrades the performance of our heuristics. We find that in Section 24, 13 out of 74 sentences contain a parsing error in the relevant aspects, but the effects are typically small and result from well-known parser issues, mainly attachment errors. As we can see in Table 3, the heuristic performance using an automatic parser degrades only slightly, and as such we can expect an automatic parser to contribute to improving RSR classification (as indeed it does).

	Pdtb	Test	Set	Auto	Test	Set
	U	**Syn**	**P**	**U**	**Syn**	**P**
Cau/Con	59.6	60.5	54.5	66.3	65.8	60.8
Cau/NR	72.2	74.9	52.6	70.3	70.2	57.3
Con/NR	61.6	60.2	52.2	69.4	69.8	56.8

Table 4: Classifier accuracy for the Unfiltered (U), Syntactically Filtered (Syn), and POS (P) models described in Section 7.3, over PDTB and Auto test sets. Baseline in all cases is 50%.

7.3 Classification Evaluation

We evaluate the impact of our syntactic heuristics on classification over the Auto and PDTB test sets using the same instance set of 400,000 training instances per relation. However, each applies different filters to the instances I before computing the frequencies F (all other parameters use the same values; these are set slightly differently than the optimized values discussed earlier because of the smaller training sets). In addition to an Unfiltered baseline, we evaluate Filtered models obtained with our syntactic heuristics for Cause and Contrast. To provide an additional point of comparison, we also evaluate the Part-of-Speech based filtering heuristic described by Marcu and Echihabi, which retains only nouns and verbs. Unlike the other filters, the POS-based filtering is applied to the NoRel instances as well as the Cause and Contrast instances. Table 4 summarizes the results of the classifying the PDTB and Auto test sets with these different models.

Before we examine the results, we note that the syntactic heuristic cuts a large portion of training data out. In terms of the total sum of frequencies in F_{cause}, i.e. the word pairs extracted from all cause instances, the syntactic filtering cuts out nearly half.

With this in mind, we see that while the syntactic filtering achieves slightly lower mean accuracy as compared to the Unfiltered baseline on the Auto test set, the pairs it does keep appear to be used more efficiently (the differences are significant). Even with this reduced training set, the syntactic heuristic improves performance in two out of three cases on the PDTB test set, including a 2.7 percent improvement for the Cause vs NoRel classifier. However, due to the small size of the PDTB test set, none of these differences is statistically significant.

We posit that bias in the Auto set may cause this difference in performance across training sets; spans in the Auto set are not true arguments of the relation in the PDTB sense, but nonetheless occur regularly with the cue phrases used in instance mining and thus are more likely to be present in the test set.

Lastly, we observe that the POS-based filtering described by M&E performs uniformly poorly. We have no explanation for this at present, given that M&E's results with this filter appear promising.

8 Conclusion

In this paper, we analyzed the problem of learning a model of rhetorical-semantic relations. Building on the work of Marcu and Echihabi, we first optimized several parameters of their model, which we found to have significant impact on classification accuracy. We then focused on the quality of the automatically-mined training examples, analyzing two techniques for data filtering. The first technique, based on automatic topic segmentation, added additional constraints on the instance mining patterns; the second used syntactic heuristics to cut out irrelevant portions of extracted training examples. While the topic-segmentation filtering approach achieves significant improvement and the best results overall, our analysis of the syntactic filtering approach indicates that refined heuristics and a larger set of parsed data can further improve those results. We would also like to experiment with combining the two approaches, i.e. by applying the syntactic heuristics to an instance set extracted using topic segmentation constraints. We conclude that our experiments show that these techniques can successfully refine RSR models and improve our ability to classify unknown relations.

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *CoNLL 2005*.

L. Carlson, D. Marcu, and M.E. Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Eurospeech 2001 Workshops*.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *ACL 1996*.

M. Galley, K.R. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *ACL 2003*.

R. Girju. 2003. Automatic detection of causal relations for question answering. In *ACL 2003 Workshops*.

Jerry R. Hobbs. 1979. Coherence and coreference. *Cognitive Science*, 3(1):67–90.

E. Hovy and E. Maier. 1993. Parsimonious or profligate: How many and which discourse structure relations? Unpublished Manuscript.

A. Knott and T. Sanders. 1998. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30(2):135–175.

M. Lapata and A. Lascarides. 2004. Inferring sentence-internal temporal relations. In *HLT 2004*.

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

D. Marcu and A. Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *ACL 2002*.

D. Marcu. 1997. *The Rhetorical Parsing, Summarization and Generation of Natural Language Texts*. Ph.D. thesis, University of Toronto, Department of Computer Science.

J. Martin. 1992. *English Text: System and Structure*. John Benjamins.

K.R. McKeown. 1985. *Text generation: Using discourse strategies and focus constraints to generate natural language text*. Cambridge University Press.

J.D. Moore and P. Wiemer-Hastings. 2003. Discourse in computational linguistics and artificial intelligence. In M.A. Gernbacher A.G. Graesser and S.R. Goldman, editors, *Handbook of Discourse Processes*, pages 439–487. Lawrence Erlbaum Associates.

M.G. Moser and J.D. Moore. 1996. Toward a synthesis of two accounts of discourse structure. *Computational Linguistics*, 22(3):409–420.

R. Prasad, E. Miltsakaki, N. Dinesh, A. Lee, A. Joshi, and B. Webber. 2006. The penn discourse treebank 1.0. annotation manual. Technical Report IRCS-06-01, University of Pennsylvania.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *HLT-NAACL 2003*.

C. Sporleder and A. Lascarides. 2005. Exploiting linguistic cues to classify rhetorical relations. In *RANLP 2005*.

C. Sporleder and A. Lascarides. To Appear. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*.

B. Wellner, J. Pustejovsky, C. Havasi, R. Sauri, and A. Rumshisky. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *SIGDial 2006*.

F. Wolf and E. Gibson. 2005. Representing discourse coherence: A corpus-based analysis. *Computational Linguistics*, 31(2):249–287.

A Unified Local and Global Model for Discourse Coherence

Micha Elsner, Joseph Austerweil, and Eugene Charniak
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{melsner,ec}@cs.brown.edu, joseph.austerweil@gmail.com

Abstract

We present a model for discourse coherence which combines the local entity-based approach of (Barzilay and Lapata, 2005) and the HMM-based content model of (Barzilay and Lee, 2004). Unlike the mixture model of (Soricut and Marcu, 2006), we learn local and global features jointly, providing a better theoretical explanation of how they are useful. As the local component of our model we adapt (Barzilay and Lapata, 2005) by relaxing independence assumptions so that it is effective when estimated generatively. Our model performs the ordering task competitively with (Soricut and Marcu, 2006), and significantly better than either of the models it is based on.

1 Introduction

Models of coherent discourse are central to several tasks in natural language processing: such models have been used in text generation (Kibble and Power, 2004) and evaluation of human-produced text in educational applications (Miltsakaki and Kukich, 2004; Higgins et al., 2004). Moreover, an accurate model can reveal information about document structure, aiding in such tasks as supervised summarization (Barzilay and Lapata, 2005).

Models of coherence tend to fall into two classes. Local models (Lapata, 2003; Barzilay and Lapata, 2005; Foltz et al., 1998) attempt to capture the generalization that adjacent sentences often have similar content, and therefore tend to contain related words. Models of this type are good at finding sentences that belong near one another in the document. However, they have trouble finding the beginning or end of the document, or recovering from sudden shifts in topic (such as occur at paragraph boundaries). Some local models also have trouble deciding which of a pair of related sentences ought to come first.

In contrast, the global HMM model of Barzilay and Lee (2004) tries to track the predictable changes in topic between sentences. This gives it a pronounced advantage in ordering sentences, since it can learn to represent beginnings, ends and boundaries as separate states. However, it has no local features; the particular words in each sentence are generated based only on the current state of the document. Since information can pass from sentence to sentence only in this restricted manner, the model sometimes fails to place sentences next to the correct neighbors.

We attempt here to unify the two approaches by constructing a model with both sentence-to-sentence dependencies providing local cues, and a hidden topic variable for global structure. Our local features are based on the entity grid model of (Barzilay and Lapata, 2005; Lapata and Barzilay, 2005). This model has previously been most successful in a conditional setting; to integrate it into our model, we first relax its independence assumptions to improve its performance when used generatively. Our global model is an HMM like that of Barzilay and Lee (2004), but with emission probabilities drawn from the entity grid. We present results for two tasks, the ordering task, on which global models usually do well, and the discrimination task, on which local models tend to outperform them. Our model improves on purely global or local approaches on both tasks.

Previous work by Soricut and Marcu (2006) has also attempted to integrate local and global features using a mixture model, with promising results. However, mixture models lack explanatory power; since each of the individual component models is known to be flawed, it is difficult to say that the combination is theoretically more sound than the parts, even if it usually works better. Moreover, since the model we describe uses a strict subset of the features used in the component models of (Soricut and Marcu, 2006), we suspect that adding it to the mixture would lead to still further improved results.

2 Naive Entity Grids

Entity grids, first described in (Lapata and Barzilay, 2005), are designed to capture some ideas of Centering Theory (Grosz et al., 1995), namely that adjacent utterances in a locally coherent discourses are likely to contain the same nouns, and that important nouns often appear in syntactically important roles such as subject or object. An entity grid represents a document as a matrix with a column for each entity, and a row for each sentence. The entry $r_{i,j}$ describes the syntactic role of entity j in sentence i: these roles are subject (**S**), object (**O**), or some other role (**X**)[1]. In addition there is a special marker (-) for nouns which do not appear at all in a given sentence. Each noun appears only once in a given row of the grid; if a noun appears multiple times, its grid symbol describes the most important of its syntactic roles: subject if possible, then object, or finally other. An example text is figure 1, whose grid is figure 2.

Nouns are also treated as salient or non-salient, another important concern of Centering Theory. We condition events involving a noun on the frequency of that noun. Unfortunately, this way of representing salience makes our model slightly deficient, since the model conditions on a particular noun occurring e.g. 2 times, but assigns nonzero probabilities to documents where it occurs 3 times. This is theo-

0 [The commercial pilot]$_{\mathbf{O}}$, [sole occupant of [the airplane]$_{\mathbf{X}}$]$_{\mathbf{X}}$, was not injured .
1 [The airplane]$_{\mathbf{O}}$ was owned and operated by [a private owner]$_{\mathbf{X}}$.
2 [Visual meteorological conditions]$_{\mathbf{S}}$ prevailed for [the personal cross country flight for which [a VFR flight plan]$_{\mathbf{O}}$ was filed]$_{\mathbf{X}}$.
3 [The flight]$_{\mathbf{S}}$ originated at [Nuevo Laredo , Mexico]$_{\mathbf{X}}$, at [approximately 1300]$_{\mathbf{X}}$.

Figure 1: A section of a document, with syntactic roles of noun phrases marked.

	0	1	2	3
PLAN	-	-	O	-
AIRPLANE	X	O	-	-
CONDITION	-	-	S	-
FLIGHT	-	-	X	S
PILOT	O	-	-	-
PROPER	-	-	-	X
OWNER	-	X	-	-
OCCUPANT	X	-	-	-

Figure 2: The entity grid for figure 1[2].

retically quite unpleasant but in comparing different orderings of the same document, it seems not to do too much damage.

Properly speaking entities may be referents of many different nouns and pronouns throughout the discourse, and both (Lapata and Barzilay, 2005) and (Barzilay and Lapata, 2005) present models which use coreference resolution systems to group nouns. We follow (Soricut and Marcu, 2006) in dropping this component of the system, and treat each head noun as having an individual single referent.

To model transitions in this entity grid model, Lapata and Barzilay (2005) takes a generative approach. First, the probability of a document is defined as $P(D) = P(S_i..S_n)$, the joint probability of all the sentences. Sentences are generated in order conditioned on all previous sentences:

$$P(D) = \prod_i P(S_i|S_{0..(i-1)}). \tag{1}$$

We make a Markov assumption of order h (in our experiments $h = 2$) to shorten the history. We represent the truncated history as $\vec{S}^h_{i-1} = S_{(i-h)}..S_{(i-1)}$.

Each sentence S_i can be split up into a set of nouns representing entities, E_i, and their corresponding syntactic roles R_i, plus a set of words which are not entities, W_i. The model treats W_i as independent of the previous sentences. For any fixed

[1]Roles are determined heuristically using trees produced by the parser of (Charniak and Johnson, 2005). Following previous work, we slightly conflate thematic and syntactic roles, marking the subject of a passive verb as **O**.

[2]The numeric token "1300" is removed in preprocessing, and "Nuevo Laredo" is marked as "PROPER".

set of sentences S_i, $\prod_i P(W_i)$ is always constant, and so cannot help in finding a coherent ordering. The probability of a sentence is therefore dependent only on the entities:

$$P(S_i|\vec{S}^h_{(i-1)}) = P(E_i, R_i|\vec{S}^h_{(i-1)}). \quad (2)$$

Next, the model assumes that each entity e_j appears in sentences and takes on syntactic roles independent of all the other entities. As we show in section 3, this assumption can be problematic. Once we assume this, however, we can simplify $P(E_i, R_i|\vec{S}^h_{(i-1)})$ by calculating for each entity whether it occurs in sentence i and if so, which role it takes. This is equivalent to predicting $r_{i,j}$. We represent the history of the specific entity e_j as $\vec{r}^h_{(i-1),j} = r_{(i-h),j}..r_{(i-1),j}$, and write:

$$P(E_i, R_i|\vec{S}^h_{(i-1)}) \approx \prod_j P(r_{i,j}|\vec{r}^h_{(i-1),j}). \quad (3)$$

For instance, in figure 2, the probability of S_3 with horizon 1 is the product of $P(\mathbf{S}|\mathbf{X})$ (for `FLIGHT`), $P(\mathbf{X}|-)$ (for `PROPER`), and likewise for each other entity, $P(-|\mathbf{O}), P(-|\mathbf{S}), P(-|-)^3$.

Although this generative approach outperforms several models in correlation with coherence ratings assigned by human judges, it suffers in comparison with later systems. Barzilay and Lapata (2005) uses the same grid representation, but treats the transition probabilities $P(r_{i,j}|\vec{r}_{i,j})$ for each document as features for input to an SVM classifier. Soricut and Marcu (2006)'s implementation of the entity-based model also uses discriminative training.

The generative model's main weakness in comparison to these conditional models is its assumption of independence between entities. In real documents, each sentence tends to contain only a few nouns, and even fewer of them can fill roles like subject and object. In other words, nouns compete with each other for the available syntactic positions in sentences; once one noun is chosen as the subject, the probability that any other will also become a subject (of a different subclause of the same sentence) is drastically lowered. Since the generative entity grid does not take this into account, it learns that in general, the probability of any given entity appearing in a specific sentence is low. Thus it generates blank sentences (those without any nouns at all) with overwhelmingly high probability.

It may not be obvious that this misallocation of probability mass also reduces the effectiveness of the generative entity grid in ordering fixed sets of sentences. However, consider the case where an entity has a history $\vec{r}^h$, and then does not appear in the next sentence. The model treats this as evidence that entities generally do not occur immediately after $\vec{r}^h$– but it may also happen that the entity was outcompeted by some other word with even more significance.

3 Relaxed Entity Grid

In this section, we relax the troublesome assumption of independence between entities, thus moving the probability distribution over documents away from blank sentences. We begin at the same point as above: sequential generation of sentences: $P(D) = \prod_i P(S_i|S_{0..(i-1)})$. We similarly separate the words into entities and non-entities, treat the non-entities as independent of the history $\vec{S}$ and omit them. We also distinguish two types of entities. Let the *known set* $K_i = e_j : e_j \in \vec{S}_{(i-1)}$, the set of all entities which have appeared before sentence i. Of the entities appearing in S_i, those in K_i are *known entities*, and those which are not are *new entities*. Since each entity in the document is new precisely once, we treat these as independent and omit them from our calculations as we did the non-entities. We return to both groups of omitted words in section 4 below when discussing our topic-based models.

To model a sentence, then, we generate the set of known entities it contains along with their syntactic roles, given the history and the known set K_i. We truncate the history, as above, with horizon h; note that this does not make the model Markovian, since the known set has no horizon. Finally, we consider only the portion of the history which relates to known nouns (since all non-known nouns have the same history - -). In all the equations below, we restrict E_i to known entities which actually appear in sentence i, and R_i to roles filled by known entities. The probability of a sentence is now:

$$P(S_i|\vec{S}^h_{(i-1)}) = P(E_i, R_i|\vec{R}^h_{(i-1)}). \quad (4)$$

We make one further simplification before beginning to approximate: we first generate the set of syntactic slots R_i which we intend to fill with known entities, and then decide which entities from the known set to select. Again, we assume independence from the history, so that the contribution of $P(R_i)$ for any ordering of a fixed set of sentences is constant and we omit it:

$$P(E_i, R_i | \vec{R}^h_{(i-1),j}) = P(E_i | R_i, \vec{R}^h_{(i-1),j}). \quad (5)$$

Estimating $P(E_i | R_i, \vec{R}^h_{(i-1),j})$ proves to be difficult, since the contexts are very sparse. To continue, we make a series of approximations. First let each role be filled individually (where $r \leftarrow e$ is the boolean indicator function "noun e fills role r"):

$$P(E_i | R_i, \vec{R}^h_{(i-1),j}) \approx \prod_{r \in R_i} P(r \leftarrow e_j | r, \vec{R}^h_{(i-1),j}). \quad (6)$$

Notice that this process can select the same noun e_j to fill multiple roles r, while the entity grid cannot represent such an occurrence. The resulting distribution is therefore slightly deficient.

Unfortunately, we are still faced with the sparse context $\vec{R}^h_{(i-1),j}$, the set of histories of all currently known nouns. It is much easier to estimate $P(r \leftarrow e_j | r, \vec{r}^h_{(i-1),j})$, where we condition only on the history of the particular noun which is chosen to fill slot r. However, in this case we do not have a proper probability distribution: i.e. the probabilities do not sum to 1. To overcome this difficulty we simply normalize by force[3]:

$$P(r \leftarrow e_j | r, \vec{R}^h_{(i-1),j}) \approx \quad (7)$$
$$\frac{P(r \leftarrow e_j | r, \vec{r}^h_{(i-1),j})}{\sum_{j \in K_i} P(r \leftarrow e_j | r, \vec{r}^h_{(i-1),j})}$$

The individual probabilities $P(r \leftarrow e_j | r, \vec{r}^h_{(i-1),j})$ are calculated by counting situations in the training documents in which a known noun has history $\vec{r}^h_{(i-1),j}$ and fills slot r in the next sentence, versus situations where the slot r exists but is filled by some other noun. Some rare contexts are still sparse, and so we smooth by adding a pseudocount of 1 for all events. Our model is expressed by equations (1),(4),(5),(6) and (7). In figure 2, the probability of S_3 with horizon 1 is now calculated as follows: the known set contains `PLAN`, `AIRPLANE`, `CONDITION`, `FLIGHT`, `PILOT`, `OWNER` and `OCCUPANT`. There is one syntactic role filled by a known noun, **S**. The probability is then calculated as $P(+|\mathbf{S}, \mathbf{X})$ (the probability of selecting a noun with history **X** to fill the role of **S**) normalized by $P(+|\mathbf{S}, \mathbf{O}) + P(+|\mathbf{S}, \mathbf{S}) + P(+|\mathbf{S}, \mathbf{X}) + 4 \times P(+|\mathbf{S}, -)$.

Like Lapata and Barzilay (2005), our relaxed model assigns low probability to sentences where nouns with important-seeming histories do not appear. However, in our model, the penalty is less severe if there are many competitor nouns. On the other hand, if the sentence contains many slots, giving the noun more opportunity to fill one of them, the penalty is proportionally greater if it does not appear.

4 Topic-Based Model

The model we describe above is a purely local one, and moreover it relies on a particular set of local features which capture the way adjacent sentences tend to share lexical choices. Its lack of any global structure makes it impossible for the model to recover at a paragraph boundary, or to accurately guess which sentence should begin a document. Its lack of lexicalization, meanwhile, renders it incapable of learning dependences between pairs of words: for instance, that a sentence discussing a crash is often followed by a casualty report.

We remedy both these problems by extending our model of document generation. Like Barzilay and Lee (2004), we learn an HMM in which each sentence has a hidden topic q_i, which is chosen conditioned on the previous state q_{i-1}. The emission model of each state is an instance of the relaxed entity grid model as described above, but in addition to conditioning on the role and history, we condition also on the state and on the particular set of lexical items $lex(K_i)$ which may be selected to fill the role: $P(r \leftarrow e_j | r, \vec{R}^h_{(i-1),j}, q_i, lex(K_i))$. This distribution is approximated as above by the normalized value of $P(r \leftarrow e_j | r, \vec{r}^h_{(i-1),j}, q_i, lex(e_j))$. However, due to our use of lexical information, even this may be too sparse for accurate estimation, so we back off by interpolating with the pre-

[3]Unfortunately this estimator is not consistent (that is, given infinite training data produced by the model, the estimated parameters do not converge to the true parameters). We are investigating maximum entropy estimation as a solution to this problem.

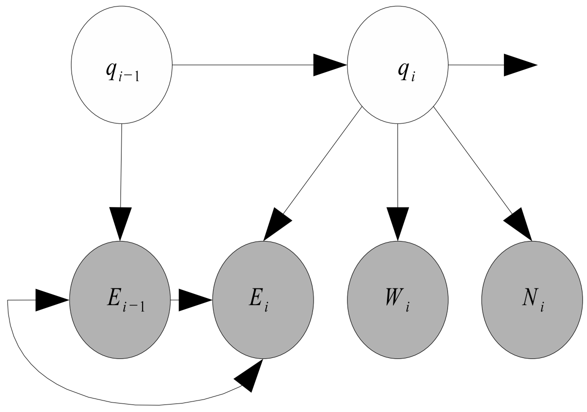


Figure 3: A single time-slice of our HMM.
$W_i \sim PY(\cdot|q_i; \theta_{LM}, discount_{LM})$
$N_i \sim PY(\cdot|q_i; \theta_{NN}, discount_{NN})$
$E_i \sim EGrid(\cdot|R, \vec{R}^2_{i-1}, q_i, lex(K_i); \theta_{EG})$
$q_i \sim DP(\cdot|q_{i-1})$
In the equations above, only the manually set interpolation hyperparameters are indicated.

vious model. In each context, we introduce θ_{EG} pseudo-observations, split fractionally according to the backoff distribution: if we abbreviate the context in the relaxed entity grid as C and the event as e, this smoothing corresponds to:

$$P(e|C, q_i, e_j) = \frac{\#(e, C, q_i, e_j) + \theta_{EG} P(e|C)}{\#(e, C, q_i, e_j) + \theta_{EG}}.$$

This is equivalent to defining the topic-based entity grid as a Dirichlet process with parameter θ_{EG} sampling from the relaxed entity grid.

In addition, we are now in a position to generate the non-entity words W_i and new entities N_i in an informative way, by conditioning on the sentence topic q_i. Since they are interrupted by the known entities, they do not form contiguous sequences of words, so we make a bag-of-words assumption. To model these sets of words, we use unigram versions of the hierarchical Pitman-Yor processes of (Teh, 2006), which implement a Bayesian version of Kneser-Ney smoothing.

To represent the HMM itself, we adapt the non-parametric HMM of (Beal et al., 2001). This is a Bayesian alternative to the conventional HMM model learned using EM, chosen mostly for convenience. Our variant of it, unlike (Beal et al., 2001), has no parameter γ to control self-transitions; our emission model is complex enough to make it unnecessary.

The actual number of states found by the model depends mostly on the backoff constants, the θs (and, for Pitman-Yor processes, $discount$s) chosen for the emission models (the entity grid, non-entity word model and new noun model), and is relatively insensitive to particular choices of prior for the other hyperparameters. As the backoff constants decrease, the emission models become more dependent on the state variable q, which leads to more states (and eventually to memorization of the training data). If instead the backoff rate increases, the emission models all become close to the general distribution and the model prefers relatively few states. We train with interpolations which generally result in around 40 states.

Once the interpolation constants are set, the model can be trained by Gibbs sampling. We also do inference over the remaining hyperparameters of the model by Metropolis sampling from uninformative priors. Convergence is generally very rapid; we obtain good results after about 10 iterations. Unlike Barzilay and Lee (2004), we do not initialize with an informative starting distribution.

When finding the probability of a test document, we do not do inference over the full Bayesian model, because the number of states, and the probability of different transitions, can change with every new observation, making dynamic programming impossible. Beal et al. (2001) proposes an inference algorithm based on particle filters, but we feel that in this case, the effects are relatively minor, so we approximate by treating the model as a standard HMM, using a fixed transition function based only on the training data. This allows us to use the conventional Viterbi algorithm. The backoff rates we choose at training time are typically too small for optimal inference in the ordering task. Before doing tests, we set them to higher values (determined to optimize ordering performance on held-out data) so that our emission distributions are properly smoothed.

5 Experiments

Our experiments use the popular AIRPLANE corpus, a collection of documents describing airplane crashes taken from the database of the National

Transportation Safety Board, used in (Barzilay and Lee, 2004; Barzilay and Lapata, 2005; Soricut and Marcu, 2006). We use the standard division of the corpus into 100 training and 100 test documents; for development purposes we did 10-fold cross-validation on the training data. The AIRPLANE documents have some advantages for coherence research: they are short (11.5 sentences on average) and quite formulaic, which makes it easy to find lexical and structural patterns. On the other hand, they do have some oddities. 46 of the training documents begin with a standard preamble: "This is preliminary information, subject to change, and may contain errors. Any errors in this report will be corrected when the final report has been completed," which essentially gives coherence models the first two sentences for free. Others, however, begin abruptly with no introductory material whatsoever, and sometimes without even providing references for their definite noun phrases; one document begins: "At V1, the DC-10-30's number 1 engine, a General Electric CF6-50C2, experienced a casing breach when the 2nd-stage low pressure turbine (LPT) anti-rotation nozzle locks failed." Even humans might have trouble identifying this sentence as the beginning of a document.

5.1 Sentence Ordering

In the sentence ordering task, (Lapata, 2003; Barzilay and Lee, 2004; Barzilay and Lapata, 2005; Soricut and Marcu, 2006), we view a document as an unordered bag of sentences and try to find the ordering of the sentences which maximizes coherence according to our model. This type of ordering process has applications in natural language generation and multi-document summarization. Unfortunately, finding the optimal ordering according to a probabilistic model with local features is NP-complete and non-approximable (Althaus et al., 2004). Moreover, since our model is not Markovian, the relaxation used as a heuristic for A^* search by Soricut and Marcu (2006) is ineffective. We therefore use simulated annealing to find a high-probability ordering, starting from a random permutation of the sentences. Our search system has few Estimated Search Errors as defined by Soricut and Marcu (2006); it rarely proposes an ordering which has lower probability than the original ordering[4].

	τ	Discr. (%)
(Barzilay and Lapata, 2005)	-	90
(Barzilay and Lee, 2004)	.44	74[5]
(Soricut and Marcu, 2006)	**.50**	-[6]
Topic-based (relaxed)	**.50**	**94**

Table 1: Results for AIRPLANE test data.

To evaluate the quality of the orderings we predict as optimal, we use *Kendall's* τ, a measurement of the number of pairwise swaps needed to transform our proposed ordering into the original document, normalized to lie between -1 (reverse order) and 1 (original order). Lapata (2006) shows that it corresponds well with human judgements of coherence and reading times. A slight problem with τ is that it does not always distinguish between proposed orderings of a document which disrupt local relationships at random, and orderings in which paragraph-like units move as a whole. In longer documents, it may be worth taking this problem into account when selecting a metric; however, the documents in the AIRPLANE corpus are mostly short and have little paragraph structure, so τ is an effective metric.

5.2 Discrimination

Our second task is the discriminative test used by (Barzilay and Lapata, 2005). In this task we generate random permutations of a test document, and measure how often the probability of a permutation is higher than that of the original document. This task bears some resemblance to the task of discriminating coherent from incoherent essays in (Miltsakaki and Kukich, 2004), and is also equivalent in the limit to the ranking metric of (Barzilay and Lee, 2004), which we cannot calculate because our model does not produce k-best output. As opposed to the ordering task, which tries to measure how *close* the model's preferred orderings are to the original, this measurement assesses how *many* orderings the model prefers. We use 20 random permutations per document, for 2000 total tests.

	τ	Discr. (%)
Naive Entity Grid	.17	81
Relaxed Entity Grid	.02	87
Topic-based (naive)	.39	85
Topic-based (relaxed)	**.54**	**96**

Table 2: Results for 10-fold cross-validation on AIRPLANE training data.

6 Results

Since the ordering task requires a model to propose the complete structure for a set of sentences, it is very dependent on global features. To perform adequately, a model must be able to locate the beginning and end of the document, and place intermediate sentences relative to these two points. Without any way of doing this, our relaxed entity grid model has τ of approximately 0, meaning its optimal orderings are essentially uncorrelated with the correct orderings[7]. The HMM content model of (Barzilay and Lee, 2004), which does have global structure, performs much better on ordering, at τ of $.44$. However, local features can help substantially for this task, since models which use them are better at placing related sentences next to one another. Using both sets of features, our topic-based model achieves state of the art performance ($\tau = .5$) on the ordering task, comparable with the mixture model of (Soricut and Marcu, 2006).

The need for good local coherence features is especially clear from the results on the discrimination task (table 1). Permuting a document may leave obvious "signposts" like the introduction and conclusion in place, but it almost always splits up many pairs of neighboring sentences, reducing local coherence. (Barzilay and Lee, 2004), which lacks local features, does quite poorly on this task (74%), while our model performs extremely well (94%).

It is also clear from the results that our relaxed entity grid model (87%) improves substantially on the generative naive entity grid (81%). When used on its own, it performs much better on the discrimination task, which is the one for which it was designed. (The naive entity grid has a higher τ score, $.17$, essentially by accident. It slightly prefers to generate infrequent nouns from the start context rather than the context - -, which happens to produce the correct placement for the "preliminary information" preamble.) When used as the emission model for known entities in our topic-based system, the relaxed entity grid shows its improved performance even more strongly (table 2); its results are about 10% higher than the naive version under both metrics.

Our combined model uses only entity-grid features and unigram language models,a strict subset of the feature set of (Soricut and Marcu, 2006). Their mixture includes an entity grid model and a version of the HMM of (Barzilay and Lee, 2004), which uses n-gram language modeling. It also uses a model of lexical generation based on the IBM-1 model for machine translation, which produces all words in the document conditioned on words from previous sentences. In contrast, we generate only entities conditioned on words from previous sentences; other words are conditionally independent given the topic variable. It seems likely therefore that using our model as a component of a mixture might improve on the state of the art result.

7 Future Work

Ordering in the AIRPLANE corpus and similar constrained sets of short documents is by no means a solved problem, but the results so far show a good deal of promise. Unfortunately, in longer and less formulaic corpora, the models, inference algorithms and even evaluation metrics used thus far may prove extremely difficult to scale up. Domains with more natural writing styles will make lexical prediction a much more difficult problem. On the other hand, the wider variety of grammatical constructions used may motivate more complex syntactic features, for instance as proposed by (Siddharthan et al., 2004) in sentence clustering.

Finding optimal orderings is a difficult task even for short documents, and will become exponentially more challenging in longer ones. For multi-paragraph documents, it is probably impractical to use full-scale coherence models to find optimal or-

[4]0 times on test data, 3 times in cross-validation.

[5]Calculated on our test permutations using the code at *http://people.csail.mit.edu/regina/code.html*.

[6]Soricut and Marcu (2006) do not report results on this task, except to say that their implementation of the entity grid performs comparably to (Barzilay and Lapata, 2005).

[7]Barzilay and Lapata (2005) do not report τ scores.

derings directly. A better approach may be a coarse-to-fine or hierarchical system which cuts up longer documents into more manageable chunks that can be ordered as a unit.

Multi-paragraph documents also pose a problem for the τ metric itself. In documents with clear thematic divisions between their different sections, a good ordering metric should treat transposed paragraphs differently than transposed sentences.

8 Acknowledgements

We are extremely grateful to Regina Barzilay, for her code, data and extensive support, Mirella Lapata for code and advice, and the BLLIP group, especially Tom Griffiths, Sharon Goldwater and Mark Johnson, for comments and criticism. We were supported by DARPA GALE contract HR0011-06-2-0001 and the Karen T. Romer Foundation. Finally we thank three anonymous reviewers for their comments.

References

Ernst Althaus, Nikiforos Karamanis, and Alexander Koller. 2004. Computing locally coherent discourses. In *Proceedings of the 42nd ACL*, Barcelona.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: an entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

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

Matthew J. Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. 2001. The infinite Hidden Markov Model. In *NIPS*, pages 577–584.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proc. of the 2005 Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 173–180.

Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *HLT-NAACL*, pages 185–192.

Roger Kibble and Richard Power. 2004. Optimising referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, pages 1085–1090.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the annual meeting of ACL, 2003.*

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):1–14.

E. Miltsakaki and K. Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Nat. Lang. Eng.*, 10(1):25–55.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *COLING04*, pages 896–902.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the Association for Computational Linguistics Conference (ACL-2006)*.

Y.W. Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, National University of Singapore.

Randomized Decoding for Selection-and-Ordering Problems

Pawan Deshpande, Regina Barzilay and David R. Karger
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{pawand,regina,karger}@csail.mit.edu

Abstract

The task of selecting and ordering information appears in multiple contexts in text generation and summarization. For instance, methods for title generation construct a headline by selecting and ordering words from the input text. In this paper, we investigate decoding methods that simultaneously optimize selection and ordering preferences. We formalize decoding as a task of finding an acyclic path in a directed weighted graph. Since the problem is NP-hard, finding an exact solution is challenging. We describe a novel decoding method based on a randomized color-coding algorithm. We prove bounds on the number of color-coding iterations necessary to guarantee any desired likelihood of finding the correct solution. Our experiments show that the randomized decoder is an appealing alternative to a range of decoding algorithms for selection-and-ordering problems, including beam search and Integer Linear Programming.

1 Introduction

The task of selecting and ordering information appears in multiple contexts in text generation and summarization. For instance, a typical multidocument summarization system creates a summary by selecting a subset of input sentences and ordering them into a coherent text. Selection and ordering at the word level is commonly employed in lexical realization. For instance, in the task of title generation, the headline is constructed by selecting and ordering words from the input text.

Decoding is an essential component of the selection-and-ordering process. Given selection and ordering preferences, the task is to find a sequence of elements that maximally satisfies these preferences. One possible approach for finding such a solution is to decompose it into two tasks: first, select a set of words based on individual selection preferences, and then order the selected units into a well-formed sequence. Although the modularity of this approach is appealing, the decisions made in the selection step cannot be retracted. Therefore, we cannot guarantee that selected units can be ordered in a meaningful way, and we may end up with a suboptimal output.

In this paper, we investigate decoding methods that simultaneously optimize selection and ordering preferences. We formalize decoding as finding a path in a directed weighted graph.[1] The vertices in the graph represent units with associated selection scores, and the edges represent pairwise ordering preferences. The desired solution is the highest-weighted acyclic path of a prespecified length. The requirement for acyclicity is essential because in a typical selection-and-ordering problem, a well-formed output does not include any repeated units. For instance, a summary of multiple documents should not contain any repeated sentences.

[1]We assume that the scoring function is local; that is, it is computed by combining pairwise scores. In fact, the majority of models that are used to guide ordering (i.e., bigrams) are local scoring functions.

Proceedings of NAACL HLT 2007, pages 444–451,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

Since the problem is NP-hard, finding an exact solution is challenging. We introduce a novel randomized decoding algorithm[2] based on the idea of color-coding (Alon et al., 1995). Although the algorithm is not guaranteed to find the optimal solution on any single run, by increasing the number of runs the algorithm can guarantee an arbitrarily high probability of success. The paper provides a theoretical analysis that establishes the connection between the required number of runs and the likelihood of finding the correct solution.

Next, we show how to find an exact solution using an integer linear programming (ILP) formulation. Although ILP is NP-hard, this method is guaranteed to compute the optimal solution. This allows us to experimentally investigate the trade-off between the accuracy and the efficiency of decoding algorithms considered in the paper.

We evaluate the accuracy of the decoding algorithms on the task of title generation. The decoding algorithms introduced in the paper are compared against beam search, a heuristic search algorithm commonly used for selection-and-ordering and other natural language processing tasks. Our experiments show that the randomized decoder is an appealing alternative to both beam search and ILP when applied to selection-and-ordering problems.

2 Problem Formulation

In this section, we formally define the decoding task for selection-and-ordering problems. First, we introduce our graph representation and show an example of its construction for multidocument summarization. (An additional example of graph construction for title generation is given in Section 6.) Then, we discuss the complexity of this task and its connection to classical NP-hard problems.

2.1 Graph Representation

We represent the set of selection units as the set of vertices V in a weighted directed graph G. The set of edges E represents pairwise ordering scores between all pairs of vertices in V. We also add a special source vertex s and sink vertex t. For each vertex v in V, we add an edge from s to v and an edge from v to t. We then define the set of all vertices as $V^* = V \cup \{s, t\}$, and the set of all edges as $E^* = E \cup \{(s, v) \; \forall \, v \in V\} \cup \{(v, t) \; \forall \, v \in V\}$.

To simplify the representation, we remove all vertex weights in our graph structure and instead shift the weight for each vertex onto its incoming edges. For each pair of distinct vertices $(v, u) \in V$, we set the weight of edge $e_{v,u}$ to be the sum of the logarithms of the selection score for u and the pairwise ordering score of (v, u).

We also enhance our graph representation by grouping sets of vertices into *equivalence classes*. We introduce these classes to control for redundancy as required in many selection-and-ordering problems.[3] For instance, in title generation, an equivalence class may consist of morphological variants of the same stem (i.e., *examine* and *examination*). Because a typical title is unlikely to contain more than one word with the same stem, we can only select a single representative from each class.

Our task is now to find the highest weighted acyclic path starting at s and ending at t with k vertices in between, such that no two vertices belong to the same equivalence class.

2.2 Example: Decoding for Multidocument Summarization

In multidocument summarization, the vertices in the decoding graph represent sentences from input documents. The vertices may be organized into equivalence classes that correspond to clusters of sentences conveying similar information. The edges in the graph represent the combination of the selection and the ordering scores. The selection scores encode the likelihood of a sentence to be extracted, while pairwise ordering scores capture coherence-based precedence likelihood. The goal of the decoder is to find the sequence of k non-redundant sentences that optimize both the selection and the ordering scores. Finding an acyclic path with the highest weight will achieve this goal.

[2]The code is available at `http://people.csail.mit.edu/pawand/rand/`

[3]An alternative approach for redundancy control would be to represent all the members of an equivalence class as a single vertex in the graph. However, such an approach does not allow us to select the best representative from the class. For instance, one element in the equivalence class may have a highly weighted incoming edge, while another may have a highly weighted outgoing edge.

2.3 Relation to Classical Problems

Our path-finding problem may seem to be similar to the tractable shortest paths problem. However, the requirement that the path be long makes it more similar to the the Traveling Salesman Problem (TSP). More precisely, our problem is an instance of the *prize collecting traveling salesman problem*, in which the salesman is required to visit k vertices at best cost (Balas, 1989; Awerbuch et al., 1995).

Since our problem is NP-hard, we might be pessimistic about finding an exact solution. But our problem has an important feature: the length k of the path we want to find is small relative to the number of vertices n. This feature distinguishes our task from other decoding problems, such as decoding in machine translation (Germann et al., 2001), that are modeled using a standard TSP formulation. In general, the connection between n and k opens up a new range of solutions. For example, if we wanted to find the best length-2 path, we could simply try all subsets of 2 vertices in the graph, in all 2 possible orders. This is a set of only $O(n^2)$ possibilities, so we can check all to identify the best in polynomial time.

This approach is very limited, however: in general, its runtime of $O(n^k)$ for paths of length k makes it prohibitive for all but the smallest values of k. We cannot really hope to avoid the exponential dependence on k, because doing so would give us a fast solution to an NP-hard problem, but there is hope of making the dependence "less exponential." This is captured by the definition of *fixed parameter tractability* (Downey and Fellows, 1995). A problem is fixed parameter tractable if we can make the exponential dependence on the parameter k *independent* of the polynomial dependence on the problem size n. This is the case for our problem: as we will describe below, an algorithm of Alon et al. can be used to achieve a running time of roughly $O(2^k n^2)$. In other words, the path length k only exponentiates a small constant, instead of the problem size n, while the dependence on n is in fact quadratic.

3 Related Work

Decoding for selection-and-ordering problems is commonly implemented using beam search (Banko et al., 2000; Corston-Oliver et al., 2002; Jin and Hauptmann, 2001). Being heuristic in nature, this algorithm is not guaranteed to find an optimal solution. However, its simplicity and time efficiency make it a decoding algorithm of choice for a wide range of NLP applications. In applications where beam decoding does not yield sufficient accuracy, researchers employ an alternative heuristic search, A* (Jelinek, 1969; Germann et al., 2001). While in some cases A* is quite effective, in other cases its running time and memory requirements may equal that of an exhaustive search. Time- and memory-bounded modifications of A* (i.e., IDA-A*) do not suffer from this limitation, but they are not guaranteed to find the exact solution. Nor do they provide bounds on the likelihood of finding the exact solution. Newly introduced methods based on local search can effectively examine large areas of a search space (Eisner and Tromble, 2006), but they still suffer from the same limitations.

As an alternative to heuristic search algorithms, researchers also employ exact methods from combinatorial optimization, in particular integer linear programming (Germann et al., 2001; Roth and Yih, 2004). While existing ILP solvers find the exact solution eventually, the running time may be too slow for practical applications.

Our randomized decoder represents an important departure from previous approaches to decoding selection-and-ordering problems. The theoretically established bounds on the performance of this algorithm enable us to explicitly control the trade-off between the quality and the efficiency of the decoding process. This property of our decoder sets it apart from existing heuristic algorithms that cannot guarantee an arbitrarily high probability of success.

4 Randomized Decoding with Color-Coding

One might hope to solve decoding with a dynamic program (like that for shortest paths) that grows an optimal path one vertex at a time. The problem is that this dynamic program may grow to include a vertex already on the path, creating a cycle. One way to prevent this is to remember the vertices used on each partial path, but this creates a dynamic program with too many states to compute efficiently.

Instead, we apply a *color coding* technique of

Alon et al (1995). The basic step of the algorithm consists of randomly coloring the graph vertices with a set of colors of size r, and using dynamic programming to find the optimum length-k path *without repeated colors*. (Later, we describe how to determine the number of colors r.) Forbidding repeated colors excludes cycles as required, but remembering only colors on the path requires less state than remembering precisely which vertices are on the path. Since we color randomly, any single iteration is not guaranteed to find the optimal path; in a given coloring, two vertices along the optimal path may be assigned the same color, in which case the optimal path will never be selected. Therefore, the whole process is repeated multiple times, increasing the likelihood of finding an optimal path.

Our algorithm is a variant of the original color-coding algorithm (Alon et al., 1995), which was developed to detect the existence of paths of length k in an unweighted graph. We modify the original algorithm to find the highest weighted path and also to handle equivalence classes of vertices. In addition, we provide a method for determining the optimal number of colors to use for finding the highest weighted path of length k.

We first describe the dynamic programming algorithm. Next, we provide a probabilistic bound on the likelihood of finding the optimal solution, and present a method for determining the optimal number of colors for a given value of k.

Dynamic Programming Recall that we began with a weighted directed graph G to which we added artificial *start* and *end* vertices s and t. We now posit a *coloring* of that graph that assigns a color c_v to each vertex v aside from s and t. Our dynamic program returns the maximum score path of length $k+2$ (including the artificial vertices s and t) from s to t with no repeated colors.

Our dynamic program grows *colorful paths*—paths with at most one vertex of each color. For a given colorful path, we define the *spectrum* of a path to be the set of colors (each used exactly once) of nodes on the *interior* of the path—we exclude the starting vertex (which will always be s) and the ending vertex. To implement the dynamic program, we maintain a table $q[v, S]$ indexed by a path-ending vertex v and a spectrum S. For vertex v and spectrum S, entry $q[v, S]$ contains the value of the maximum-score colorful path that starts at s, terminates at v, and has spectrum S in its interior.

We initialize the table with length-one paths: $q[v, \emptyset]$ represents the path from s to v, whose spectrum is the empty set since there are no interior vertices. Its value is set to the score of edge (s, v). We then iterate the dynamic program k times in order to build paths of length $k + 1$ starting at s. We observe that the optimum colorful path of length ℓ and spectrum S from s to v must consist of an optimum path from s to u (which will already have been found by the dynamic program) concatenated to the edge (u, v). When we concatenate (u, v), vertex u becomes an interior vertex of the path, and so its color must not be in the preexisting path's spectrum, but joins the spectrum of the path we build. It follows that

$$q[v, S] = \max_{(u,v) \in G, c_u \in S, c_v \notin S} q[u, S - \{c_u\}] + w(u, v)$$

After k iterations, for each vertex v we will have a list of optimal paths from s to v of length $k + 1$ with all possible spectra. The optimum length-$k+2$ colorful path from s to t must follow the optimum length-$k+1$ path of *some* spectrum to *some* penultimate vertex v and then proceed to vertex t; we find it by iterating over all such possible spectra and all vertices v to determine $argmax_{v,S} q[v, S] + w(v, t)$.

Amplification The algorithm of Alon et al., and the variant we describe, are somewhat peculiar in that the probability of finding the optimal solution in one coloring iteration is quite small. But this can easily be dealt with using a standard technique called *amplification* (Motwani and Raghavan, 1995). Suppose that the algorithm succeeds with small probability p, but that we would like it to succeed with probability $1 - \delta$ where δ is very small. We run the algorithm $t = (1/p) \ln 1/\delta$ times. The probability that the algorithm fails every single run is then $(1 - p)^t \leq e^{-pt} = \delta$. But if the algorithm succeeds on even one run, then we will find the optimum answer (by taking the best of the answers we see).

No matter how many times we run the algorithm, we cannot absolutely guarantee an optimal answer. However, the chance of failure can easily be driven to negligible levels—achieving, say, a one-in-a-billion chance of failure requires only $20/p$ iterations by the previous analysis.

Determining the number of colors Suppose that we use r random colors and want to achieve a given failure probability δ. The probability that the optimal path has no repeated colors is:

$$1 \cdot \frac{r-1}{r} \cdot \frac{r-2}{r} \cdots \frac{r-(k-1)}{r}.$$

By the amplification analysis, the number of trials needed to drive the failure probability to the desired level will be inversely proportional to this quantity. At the same time, the dynamic programming table at each vertex will have size 2^r (indexing on a bit vector of colors used per path), and the runtime of each trial will be proportional to this. Thus, the running time for the necessary number of trials T_r will be proportional to

$$1 \cdot \frac{r}{r-1} \cdot \frac{r}{r-2} \cdots \frac{r}{r-(k-1)} \cdot 2^r$$

What $r \geq k$ should we choose to minimize this quantity? To answer, let us consider the ratio:

$$\begin{aligned} \frac{T_{r+1}}{T_r} &= \left(\frac{r+1}{r}\right)^k \cdot \frac{r-(k-1)}{r+1} \cdot 2 \\ &= 2(1+1/r)^k(1-k/(r+1)) \end{aligned}$$

If this ratio is less than 1, then using $r+1$ colors will be faster than using r; otherwise it will be slower. When r is very close to k, the above equation is tiny, indicating that one should increase r. When $r \gg k$, the above equation is huge, indicating one should decrease r. Somewhere in between, the ratio passes through 1, indicating the optimum point where neither increasing nor decreasing r will help. If we write $\alpha = k/r$, and consider large k, then $\frac{T_{r+1}}{T_r}$ converges to $2e^{\alpha}(1-\alpha)$. Solving numerically to find where this is equal to 1, we find $\alpha \approx .76804$, which yields a running time proportional to approximately $(4.5)^k$.

In practice, rather than using an (approximate) formula for the optimum r, one should simply plug all values of r in the range $[k, 2k]$ into the running-time formula in order to determine the best; doing so takes negligible time.

5 Decoding with Integer Linear Programming

In this section, we show how to formulate the selection-and-ordering problem in the ILP framework. We represent each edge (i, j) from vertex i to vertex j with an indicator variable $I_{i,j}$ that is set to 1 if the edge is selected for the optimal path and 0 otherwise. In addition, the associated weight of the edge is represented by a constant $w_{i,j}$.

The objective is then to maximize the following sum:

$$\max_I \sum_{i \in V} \sum_{j \in V} w_{i,j} I_{i,j} \tag{1}$$

This sum combines the weights of edges selected to be on the optimal path.

To ensure that the selected edges form a valid acyclic path starting at s and ending at t, we introduce the following constraints:

Source-Sink Constraints Exactly one edge originating at source s is selected:

$$\sum_{j \in V} I_{s,j} = 1 \tag{2}$$

Exactly one edge ending at sink t is selected:

$$\sum_{i \in V} I_{i,t} = 1 \tag{3}$$

Length Constraint Exactly $k+1$ edges are selected:

$$\sum_{i \in V} \sum_{j \in V} I_{i,j} = k+1 \tag{4}$$

The $k+1$ selected edges connect $k+2$ vertices including s and t.

Balance Constraints Every vertex $v \in V$ has in-degree equal to its out-degree:

$$\sum_{i \in V} I_{i,v} = \sum_{i \in V} I_{v,j} \quad \forall\, v \in V^* \tag{5}$$

Note that with this constraint, a vertex can have at most one outgoing and one incoming edge.

Redundancy Constraints To control for redundancy, we require that at most one representative from each equivalence class is selected. Let Z be a set of vertices that belong to the same equivalence class. For every equivalence class Z, we force the total out-degree of all vertices in Z to be at most 1.

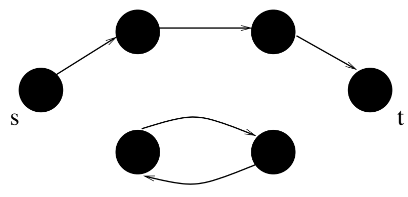


Figure 1: A subgraph that contains a cycle, while satisfying constraints 2 through 5.

$$\sum_{i \in Z} \sum_{j \in V} I_{i,j} \leq 1 \quad \forall\, Z \subseteq V \tag{6}$$

Acyclicity Constraints The constraints introduced above do not fully prohibit the presence of cycles in the selected subgraph. Figure 1 shows an example of a selected subgraph that contains a cycle while satisfying all the above constraints.

We force acyclicity with an additional set of variables. The variables $f_{i,j}$ are intended to number the edges on the path from 1 to $k+1$, with the first edge getting number $f_{i,j} = k+1$, and the last getting number $f_{i,j} = 1$. All other edges will get $f_{i,j} = 0$. To enforce this, we start by ensuring that only the edges selected for the path ($I_{i,j} = 1$) get nonzero f-values:

$$0 \leq f_{i,j} \leq (k+1)\, I_{i,j} \quad \forall\, i, j \in V \tag{7}$$

When $I_{i,j} = 0$, this constraint forces $f_{i,j} = 0$. When $I_{i,j} = 1$, this allows $0 \leq f_{i,j} \leq k{+}1$. Now we introduce additional variables and constraints. We constrain demand variables d_v by:

$$d_v \;=\; \sum_{i \in V} I_{i,v} \quad \forall\, v \in V^* - \{s\} \tag{8}$$

The right hand side sums the number of selected edges entering v, and will therefore be either 0 or 1. Next we add variables a_v and b_v constrained by the equations:

$$a_v \;=\; \sum_{i \in V} f_{i,v} \tag{9}$$

$$b_v \;=\; \sum_{i \in V} f_{v,i} \tag{10}$$

Note that a_v sums over f values on all edges entering v. However, by the previous constraints those f-values can only be nonzero on the (at most one) selected edge entering v. So, a_v is simply the f-value on the selected edge entering v, if one exists, and 0 otherwise. Similarly, b_v is the f-value on the (at most one) selected edge leaving v.

Finally, we add the constraints

$$a_v - b_v \;=\; d_v \quad v \neq s \tag{11}$$

$$b_s \;=\; k+1 \tag{12}$$

$$a_t \;=\; 1 \tag{13}$$

These last constraints let us argue, by induction, that a path of length exactly $k+1$ must run from s to t, as follows. The previous constraints forced exactly one edge leaving s, to some vertex v, to be selected. The constraint $b_s = k+1$ means that the f-value on this edge must be $k+1$. The balance constraint on v means some edge must be selected leaving v. The constraint $a_v - b_v = d_v$ means this edge must have f-value k. The argument continues the same way, building up a path. The balance constraints mean that the path must terminate at t, and the constraint that $a_t = 1$ forces that termination to happen after exactly $k+1$ edges.[4]

For those familiar with max-flow, our program can be understood as follows. The variables I force a flow, of value 1, from s to t. The variables f represent a flow with supply $k+1$ at s and demand d_v at v, being forced to obey "capacity constraints" that let the flow travel only along edges with $I = 1$.

6 Experimental Set-Up

Task We applied our decoding algorithm to the task of title generation. This task has been extensively studied over the last six years (Banko et al., 2000; Jin and Hauptmann, 2001). Title generation is a classic selection-and-ordering problem: during title realization, an algorithm has to take into account both the likelihood of words appearing in the title and their ordering preferences. In the previous approaches, beam search has been used for decoding. Therefore, it is natural to explore more sophisticated decoding techniques like the ones described in this paper.

Our method for estimation of selection-and-ordering preferences is based on the technique described in (Banko et al., 2000). We compute the

[4]The network flow constraints allow us to remove the previously placed length constraint.

likelihood of a word in the document appearing in the title using a maximum entropy classifier. Every stem is represented by commonly used positional and distributional features, such as location of the first sentence that contains the stem and its TF*IDF. We estimate the ordering preferences using a bigram language model with Good-Turing smoothing.

In previous systems, the title length is either provided to a decoder as a parameter, or heuristics are used to determine it. Since exploration of these heuristics is not the focus of our paper, we provide the decoder with the actual title length (as measured by the number of content words).

Graph Construction We construct a decoding graph in the following fashion. Every unique content word comprises a vertex in the graph. All the morphological variants of a stem belong to the same equivalence class. An edge (v, u) in the graph encodes the selection preference of u and the likelihood of the transition from v to u.

Note that the graph does not contain any auxiliary words in its vertices. We handle the insertion of auxiliary words by inserting additional edges. For every auxiliary word x, we add one edge representing the transition from v to u via x, and the selection preference of u. The auxiliary word set consists of 24 prepositions and articles extracted from the corpus.

Corpus Our corpus consists of 547 sections of a commonly used undergraduate algorithms textbook. The average section contains 609.2 words. A title, on average, contains 3.7 words, among which 3.0 are content words; the shortest and longest titles have 1 and 13 words respectively. Our training set consists of the first 382 sections, the remaining 165 sections are used for testing. The bigram language model is estimated from the body text of all sections in the corpus, consisting of 461,351 tokens.

To assess the importance of the acyclicity constraint, we compute the number of titles that have repeated content words. The empirical findings support our assumption: 97.9% of the titles do not contain repeated words.

Decoding Algorithms We consider three decoding algorithms: our color-coding algorithm, ILP, and beam search.[5] The beam search algorithm can only consider vertices which are not already in the path.[6]

To solve the ILP formulations, we employ a Mixed Integer Programming solver *lp_solve* which implements the Branch-and-Bound algorithm. We implemented the rest of the decoders in Python with the Psyco speed-up module. We put substantial effort to optimize the performance of all of the algorithms. The color-coding algorithm is implemented using parallelized computation of coloring iterations.

7 Results

Table 1 shows the performance of various decoding algorithms considered in the paper. We first evaluate each algorithm by the running times it requires to find all the optimal solutions on the test set. Since ILP is guaranteed to find the optimal solution, we can use its output as a point of comparison. Table 1 lists both the average and the median running times. For some of the decoding algorithms, the difference between the two measurements is striking — 6,536 seconds versus 57.3 seconds for ILP. This gap can be explained by outliers which greatly increase the average running time. For instance, in the worst case, ILP takes an astounding 136 hours to find the optimal solution. Therefore, we base our comparison on the median running time.

The color-coding algorithm requires a median time of 9.7 seconds to find an optimal solution compared to the 57.3 seconds taken by ILP. Furthermore, as Figure 2 shows, the algorithm converges quickly: just eleven iterations are required to find an optimal solution in 90% of the titles, and within 35 iterations all of the solutions are found. An alternative method for finding optimal solutions is to employ a beam search with a large beam size. We found that for our test set, the smallest beam size that satisfies this condition is 1345, making it twenty-three times slower than the randomized decoder with respect to the median running time.

Does the decoding accuracy impact the quality of the generated titles? We can always trade speed for accuracy in heuristic search algorithms. As an extreme, consider a beam search with a beam of size 1: while it is very fast with a median running time

[5]The combination of the acyclicity and path length constraints require an exponential number of states for A* since each state has to preserve the history information. This prevents us from applying A* to this problem.

[6]Similarly, we avoid redundancy by disallowing two vertices from the same equivalence class to belong to the same path.

	Average (s)	Median (s)	ROUGE-L	Optimal Solutions (%)
Beam 1	0.6	0.4	0.0234	0.0
Beam 80	28.4	19.3	0.2373	64.8
Beam 1345	368.6	224.4	0.2556	100.0
ILP	6,536.2	57.3	0.2556	100.0
Color-coding	**73.8**	**9.7**	**0.2556**	**100.0**

Table 1: Running times in seconds, ROUGE scores, and percentage of optimal solutions found for each of the decoding algorithms.

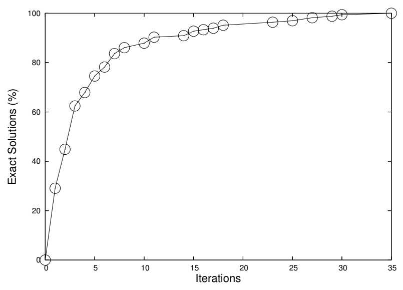


Figure 2: The proportion of exact solutions found for each iteration of the color coding algorithm.

of less than one second, it is unable to find any of the optimal solutions. The titles generated by this method have substantially lower scores than those produced by the optimal decoder, yielding a 0.2322 point difference in ROUGE scores. Even a larger beam size such as 80 (as used by Banko et al. (2000)) does not match the title quality of the optimal decoder.

8 Conclusions

In this paper, we formalized the decoding task for selection-and-ordering as a problem of finding the highest-weighted acyclic path in a directed graph. The presented decoding algorithm employs randomized color-coding, and can closely approximate the ILP performance, without blowing up the running time. The algorithm has been tested on title generation, but the decoder is not specific to this task and can be applied to other generation and summarization applications.

9 Acknowledgements

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865). We also would like to acknowledge the MIT NLP group and the anonymous reviewers for valuable comments.

References

N. Alon, R. Yuster, U. Zwick. 1995. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856.

B. Awerbuch, Y. Azar, A. Blum, S. Vempala. 1995. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proceedings of the STOC*, 277–283.

E. Balas. 1989. The prize collecting traveling salesman problem. *Networks*, 19:621–636.

M. Banko, V. O. Mittal, M. J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the ACL*, 318–325.

S. Corston-Oliver, M. Gamon, E. Ringger, R. Moore. 2002. An overview of amalgam: A machine-learned generation module. In *Proceedings of INLG*, 33–40.

R. G. Downey, M. R. Fellows. 1995. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theoretical Computer Science*, 141(1–2):109–131.

J. Eisner, R. W. Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *Proceedings of the HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

U. Germann, M. Jahr, K. Knight, D. Marcu, K. Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the EACL/ACL*, 228–235.

F. Jelinek. 1969. A fast sequential decoding algorithm using a stack. *IBM Research Journal of Research and Development*.

R. Jin, A. G. Hauptmann. 2001. Automatic title generation for spoken broadcast news. In *Proceedings of the HLT*, 1–3.

R. Motwani, P. Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press, New York, NY.

D. Roth, W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the CONLL*, 1–8.

Multilingual Structural Projection across Interlinear Text

Fei Xia
Department of Linguistics
University of Washington
Seattle, WA 98195
fxia@u.washington.edu

William D. Lewis
Department of Linguistics
University of Washington
Seattle, WA 98195
wlewis2@u.washington.edu

Abstract

This paper explores the potential for annotating and enriching data for low-density languages via the alignment and projection of syntactic structure from parsed data for resource-rich languages such as English. We seek to develop enriched resources for a large number of the world's languages, most of which have no significant digital presence. We do this by tapping the body of Web-based linguistic data, most of which exists in small, analyzed chunks embedded in scholarly papers, journal articles, Web pages, and other online documents. By harvesting and enriching these data, we can provide the means for knowledge discovery across the resulting corpus that can lead to building computational resources such as grammars and transfer rules, which, in turn, can be used as bootstraps for building additional tools and resources for the languages represented.[1]

1 Introduction

Developing natural language applications is generally dependent on the availability of annotated corpora. Building annotated resources, however, is a significantly time consuming process involving considerable human effort. Although a number of projects have been undertaken to develop annotated resources for non-English languages, e.g., treebanks, the development of these resources has been no small feat, and to date have been limited to a very small number of the world's languages (e.g., Chinese, German, Arabic, Korean, etc.). Some notable efforts have been undertaken to develop automated means for creating annotated corpora through the projection of annotations (Yarowksy and Ngai, 2001; Xi and Hwa, 2005). The resulting methods, however, can only be applied to a small number of language pairs due mostly to the need for sizeable parallel corpora. Unfortunately, most languages do not have parallel corpora of sufficient size, making these methods inapplicable for the vast majority of the world's languages.

We describe a method for bootstrapping resource creation by tapping the wealth of multilingual data on the Web that has been created by linguists. Of particular note is the linguistic presentation format of "interlinear text", a common format used for presenting language data and analysis relevant to a particular argument or investigation. Since interlinear examples consist of orthographically or phonetically encoded language data aligned with an English translation, the "database" of interlinear examples found on the Web, when taken together, constitute a significant multilingual, parallel corpus covering hundreds to thousands of the world's languages.

We do not propose that a database of interlinear text alone is sufficient to create NLP resources and tools, but rather that it may act as a means for more rapidly developing such tools using less data. We contend that such a resource allows one to develop computational artifacts, such as grammars and transfer rules, which can be used as "seed" knowledge for building larger resources. In particular, knowing a little about the structure of a language can help in developing annotated corpora and tools, since a little knowledge can go a long way in inducing accurate structure and annotations (Haghighi and Klein, 2006).

Of particular relevance to MT is the issue of struc-

[1]We would like to thank Dan Jinguji for creating the word alignment and source dependency structure gold standards. Our thanks also go to three anonymous reviewers for their helpful comments and suggestions.

tural divergence (Dorr, 1994). Many MT models implicitly make the so-called direct correspondence assumption (DCA) as defined in (Hwa et al., 2002). However, to what extent that assumption holds is tested only on a small number of language pairs using hand aligned data (Fox, 2002; Hwa et al., 2002; Wellington et al., 2006). A larger sample of typologically diverse language data can help test the assumption for hundreds of languages.

We contend that the knowledge garnered from structural projections applied to interlinear text can bootstrap the development of resources and tools across parallel corpora, where such corpora could be of smaller size and the resulting tools more robust, opening the door to the development of tools and resources for a larger number of the world's languages. Given the imminent death of half of the world's 6,000 languages (Krauss, 1992), the development of *any* language specific tools for a larger percentage of the world's languages than is currently possible can aid in both their documentation and preservation.

2 Background

The practice of presenting language data in interlinear form has a long history in the field of linguistics, going back at least to the time of the structuralists (see (Swanton, 1912) for early examples). The modern form of interlinear data presentation started to gel in the mid-1960s, resulting in the canonical three line form shown in Ex (1), which we will refer to as Interlinear Glossed Text, or IGT. The canonical form consists of three lines: a line for the language in question (often a sentence, which we will refer to here as the *source sentence*), an English gloss line, and an English translation.[2]

(1)	Rhoddodd	yr	athro	lyfr	i'r	bachgen	ddoe
	gave-3sg	the	teacher	book	to-the	boy	yesterday

"The teacher gave a book to the boy yesterday"
(Bailyn, 2001)

Although IGT is usually embedded in linguistics documents as part of a larger analysis, in and of itself it contains analysis and interesting information about the source language. In particular, the gloss line, which is word and morpheme aligned with the source, contains word and morpheme translations for the source language data, and can even contain grammatically salient annotations (e.g., 3sg for Third Person Singular). Further, the reader will note that many words are shared between the gloss and translation lines, allowing for the alignment between these two lines as a intermediate step in the alignment between the translation and the source.

An effort is underway to collect these interlinear snippets into an online searchable database, the primary purpose of which is to help linguists find analyzed data for languages they are interested in. We use this resource, called ODIN, the Online Database of INterlinear text (Lewis, 2006)[3], as our primary data source. At the time of this writing, ODIN contains 36,439 instances of interlinear data for 725 of the world's languages.

3 The Enrichment Algorithm

Our algorithm enriches the original IGT examples by building syntactic structures over the English data and then projects these onto the source language data via word alignment. The term *syntactic structure* in this paper refers to both phrase structure (PS) and dependency structure (DS). The enrichment process has three steps:

1. Parse the English translation using an off-the-shelf parser.
2. Align the source sentence and English translation with the help of the gloss line.
3. Project the English syntactic structures to obtain the source syntactic structures using word alignment.

3.1 Parsing English sentences

There are many English parsers available to the public, and in this experiment we used Charniak's parser (Charniak, 1997), which was trained on the English Penn Treebank (Marcus et al., 1994). Figure 1(a) shows a parse tree (in the Penn Treebank style) for the English translation in Ex (1). Given a parse tree, we use a head percolation table (Magerman, 1995) to create the corresponding dependency structure. Figure 2(a) shows the dependency structure derived from the parse tree in Figure 1(a).

3.2 Word alignment

Because most of the 700+ languages in ODIN are low-density languages with no on-line bilingual dictionaries or large parallel corpora, aligning the source sentence and its English translation *directly* would not work well. To take advantage of the unique layout of IGT examples, we propose using the gloss line as a bridge between the other two lines; that is, we first align the source sentence and the gloss line, and then align the gloss line and the English translation. The process is illustrated in Figure 3.

[2]As pointed out by a reviewer, there is a long tradition in the classical languages for using interlinear translations. So, too, in other literature bases. Our focus here is strictly limited to IGT, the interlinear form used in the field of linguistics.

[3]The url of ODIN is http://www.csufresno.edu/odin

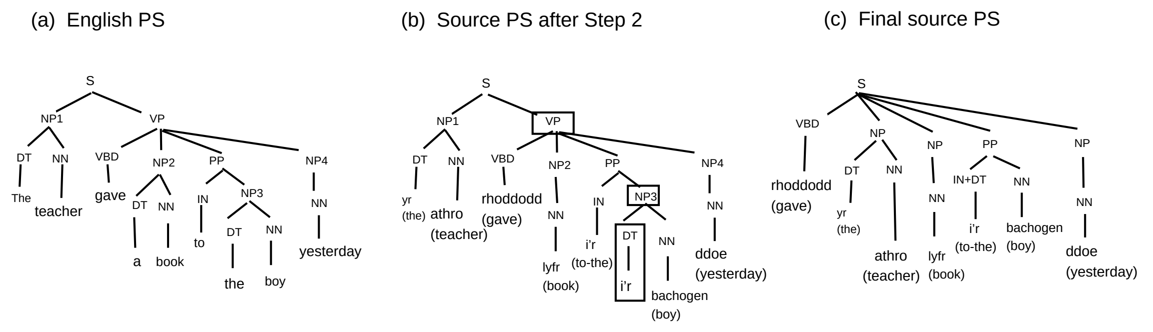


Figure 1: English PS produced by Charniak's parser, and source PS projected from the English PS

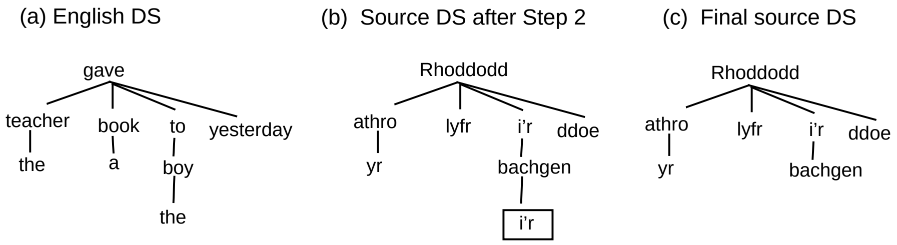


Figure 2: English DS derived from English PS, and source DS projected from the English DS

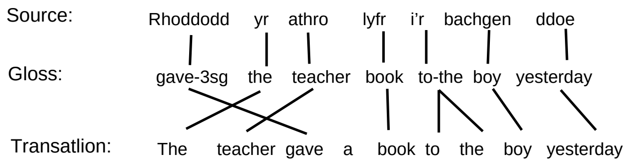

Source:
Rhoddodd yr athro lyfr i'r bachgen ddoe
Gloss:
gave-3sg the teacher book to-the boy yesterday
Transatlion:
The teacher gave a book to the boy yesterday

Figure 3: Aligning source sentence and English translation with help of the gloss line

The alignment between the source sentence and the gloss line is trivial and our preliminary experiments showed that simply using whitespace and dashes as delimiters, and assuming a one-to-one alignment produces almost perfect results. In contrast, the alignment between the gloss line and the English translation is more complicated since alignment links can cross and words on one side can link to zero or more words on the other side. We built two aligners for this stage, as described below.

3.2.1 Statistical word aligner

We create a parallel corpus by using the gloss lines and the translation lines of all the IGT examples for all the languages in ODIN. We then train IBM models (Brown et al., 1993) using the GIZA++ package (Och and Ney, 2000). In addition to the common practice of lowercasing words and combining word alignments from both directions, we adopt the following strategies to improve word alignment:

Breaking words into morphemes: Since a multi-morpheme word in a gloss line often corresponds to multiple words in the translation line, we split each word on the gloss line into morphemes using the standard IGT morpheme delimiters (*e.g.*, "-"). For instance, the seven words in the gloss line of Ex (1) become nine morphemes.

Adding (x,x) pairs: If a word x appears in the gloss and the translation lines of the same IGT example, it is highly likely that the two copies of the same word should be aligned to each other. To help GIZA++ recognize this property, we first identify and collect all such words and then add single word pairs (x,x) to the training data. For instance, from Ex (1), we would add a sentence pair for each morpheme (excepting *-3sg* which does not appear in the translation line).

3.2.2 Heuristic word aligner

Our second word aligner is based on the assumption that if two words (one on the gloss line, the other on the translation line) have the same root form, they are likely to be aligned to one other.We built a simple English morphological analyzer and ran it on the two lines, and then linked the words with the same root form.[4]

3.3 Tree projection

We designed two projection algorithms: one which projects PS and the other which projects DS, both from the English to the source language.[5]

3.3.1 Projecting dependency structure

Our DS projection algorithm is similar to the projection algorithms described in (Hwa et al., 2002) and (Quirk et al., 2005). It has four steps: First, we copy the English DS, and remove all the unaligned English words from the DS.[6] Second, we replace each English word in the DS with the corresponding source words. If an English word x aligns to several source words, we will make several copies of the node for x, one copy for each such source word. The copies will all be siblings in the DS.

If a source word aligns to multiple English words, after Step 2 the source word will have several copies in the resulting DS. In the third step, we keep only the copy that is closest to the root and remove all the other copies.[7] In Step 4, we attach unaligned source words to the DS using the heuristics described in (Quirk et al., 2005). Figure 2 shows the English DS, the source DS after Step 2, and the final DS.

3.3.2 Projecting phrase structure

Our PS projection algorithm also has four steps, the first two being the same as those for projecting DS. In the third step, starting from the root of the current source PS and for each node x with more than one child, we reorder each pair of x's children until they are in the same order as dictated by the source sentence. Let y_i and y_j be two children of x, and their spans be $S_i = [a_i, b_i]$ and $S_j = [a_j, b_j]$. When we reorder y_i and y_j, there are four possible scenarios:

(1) **S_i and S_j don't overlap:** we put y_i before y_j if $a_i < a_j$ or the opposity if $a_i > a_j$.

(2) **S_i is a strict subset of S_j:** we remove y_j from the PS and *promote* its children: y_j's children will become children of y_j's parent.

(3) **S_j is a strict subset of S_i:** we remove y_i and promote its children.

(4) **S_i and S_j overlap but neither is a strict subset of the other:** we remove both y_i and y_j and promote their children. If both y_i and y_j are leaf nodes with the same span, we will merge the two nodes.[8]

The last step is to insert unaligned source words into the source PS. For each unaligned source word x, we will find its closest left and right neighbors that are aligned to some English words, and then attach x to the lowest common ancestor of the two neighbors. Figure 1 shows the English PS, the source PS after Step 2, and the final source PS. The three boxes in 1(b) mark the nodes that are removed in Step 3.

4 Experiments

We tested the feasibility of our approach on a small set of IGT examples for seven languages: German (GER), Korean (KKN), Hausa (HUA), Malagasy (MEX), Welsh (WLS), Irish (GLI), and Yaqui (YAQ). This set of languages was chosen because of its typological diversity: GER and HUA are SVO languages, KKN and YAQ are SOV, GLI and WLS are VSO, and MEX is VOS. In addition, while German and Korean are well-studied and have readily accessible resources that we could use to test the effectiveness and accuracy of our methods, Yaqui, with about 16,000 speakers, is a highly endangered language and serves as a demonstration of our methods for resource-poor and endangered languages.

4.1 Creating the gold standard for the test set

The number of IGT examples in ODIN varies greatly across the seven languages, ranging from less than one hundred for Welsh to over seventeen hundred for German. For each language, we randomly picked 50-150 IGT examples from the available examples whose English translations had at least five words.[9] The examples were manually checked and corrupted examples were thrown away. The remaining examples

[4]When a word is repeated in both the gloss and translation, the individual occurrences are aligned individually in left-to-right order.

[5]The DS projection algorithm as described does not guarantee that the yield of the resulting source DS has the same word order as the source sentence; however, if needed, the algorithm can be easily modified (by making its Step 3 similar to the Step 3 of the PS projection algorithm) to ensure the correct word order.

[6]Every time we remove an internal node x from a DS, we make x's children depend on x's parent directly.

[7]The heuristic is not as arbitrary as it sounds because very often when a source word aligns to multiple English words, one of the English words dominates the rest in the DS (e.g., the node for *to* in Figure 2(a) dominates the node for *the*). We are using the dominant word to represent the whole set.

[8]We will keep one copy and merge the POS tag of the words. For instance, the tag IN+DT in Figure 1(c) was created when two copies of *i'r* in Figure 1(b) were merged.

[9]We skipped examples with very short English translations because they are unlikely to contain much in the way of syntactic structures.

Table 1: The size and average sentence length of the test data

	GER	KKN	HUA	MEX	WLS	GLI	YAQ	Total
# of IGT examples	104	103	77	87	53	46	68	538
# of src words	739	526	441	498	313	252	404	3173
Ave src sent leng	7.11	5.11	5.73	5.72	5.91	5.48	5.94	5.90
# of Eng words	711	735	520	646	329	278	544	3823
Ave Eng sent leng	7.41	7.14	6.75	7.43	6.21	6.04	8.01	7.11
# of speakers	128M	78M	39M	9.4M	580K	260K	16K	255.3M

formed our test data. Table 1 shows the size and average sentence lengths of the test data by language.[10] The languages are sorted by number of speakers (as derived from the Ethnologue (Gordon, 2005)).

We ran our algorithm on the test data, and the system produced the following: an English PS, English DS, word alignment, projected source PS, and projected source DS. We asked human annotators to manually check the output and correct the English DS, word alignments and projected DS structures where necessary.[11] [12] In order to calculate inter-annotator agreement, the Yaqui data and half of the German data were each checked by two annotators, and the disagreement between the annotators was adjudicated and a gold standard was created. The inter-annotator agreement (a.k.a. the F-measure of dependency or alignment links) on English DS, gloss-translation alignment, and projected source DS are 96.34%, 96.35%, and 91.09%, respectively. The rest of the data were annotated by one annotator.

4.2 Word alignment results

We tested our word aligners on 70% (374 examples) of the whole test set (538 examples), while reserving the remaining 30% for future use.

4.2.1 Statistical word aligner

As indicated earlier, the ODIN database contains 36,439 IGT examples. We removed duplicates[13] and examples with missing lines, and used the remaining 28,902 examples for GIZA++ training.[14] Table 2 shows the statistics of the training data with all words lowercased. Tables 3–5 show the performance of the word aligner under three settings:

Table 2: The training data for GIZA++

# of sentences	28,902
# of words in gloss lines	174,765
# of morphemes in gloss lines	251,465
# of words in translation lines	217,022
Size of gloss word vocabulary	16360
Size of gloss morpheme vocabulary	14050
Size of translation word vocabulary	14029

Table 3: The word alignment results when gloss words are not split into morphemes

	Precision	Recall	F-measure
Gloss → trans	0.674	0.689	0.681
Trans → gloss	0.721	0.823	0.769
Intersection	0.948	0.620	0.750
Union	0.590	0.892	0.711
Refined	0.846	0.780	**0.812**

(1): Not splitting words in the gloss lines into morphemes.

(2): Splitting words in gloss lines into morphemes.

(3): Doing (2) plus adding (x,x) sentence pairs into the training data, where x is a word that appears in both the gloss and translation lines of the same IGT example.

For each setting, we trained in both directions and combined the two alignments by taking the intersection, union, and refined as defined in (Och and Ney, 2000). The best F-score for each setting is in boldface. From the tables, it is clear that the third setting works the best, and combining the alignments

[10]There are three reasons why the sentences are so short. First, since IGT is used to present particular linguistically salient morphological or syntactic material, sentences in IGT are only as long as needed for the given exposé. Second, space constraints often dictate using shorter examples (i.e., they must fit on one line). Third, the IGT extraction algorithm currently used in ODIN does not search for the less common multi-line (*i.e.*, greater than three line) examples.

[11]The English PS and source PS were not corrected; without a thorough linguistic study of the source languages, it is impossible to devise appropriate *gold standards* for their phrase structures.

[12]The DS structures for the English and source language in the gold standard can be non-isomorphic.

[13]Duplicates are common since it is standard practice in linguistics to copy and cite language examples from other papers.

[14]Interestingly, although the IGT examples in the training data come from hundreds of languages in ODIN, IBM Model 4 performs significantly better than Models 1 and 2 (by at least two percent points for F-measure); therefore, all the GIZA++ results reported in the paper are based on Model 4.

Table 4: The word alignment results when gloss words are split into morphemes

	Precision	Recall	F-measure
Gloss → trans	0.746	0.889	0.811
Trans → gloss	0.797	0.863	0.829
Intersection	0.958	0.811	0.878
Union	0.659	0.941	0.775
Refined	0.918	0.900	**0.909**

Table 5: The word alignment results when (x,x) pairs are added

	Precision	Recall	F-measure
Gloss → trans	0.759	0.922	0.833
Trans → gloss	0.801	0.924	0.858
Intersection	0.956	0.885	**0.919**
Union	0.666	0.961	0.787
Refined	0.908	0.921	0.915

from both directions works better than either direction alone.[15]

4.2.2 Heuristic word aligner

The word aligner has two settings. In the first one, the aligner aligns two words if and only if they have the same *orthographic form*. In the second, it aligns two words if and only if they have the same *root* form.[16] The results are shown in the first and second rows of Table 6.

We experimented with various methods of combining the two aligners, and the best one is an augmented heuristic word aligner which links two words if and only if they have the same root form or they are *good* translations of each other according to the translation model built by GIZA++.[17] The result is shown in the last row of Table 6. We used this aligner for the structural projection experiment.

[15]For languages with hundreds of IGT examples, one may wonder whether training GIZA++ with the data for that language alone would outperform the system trained with IGT examples from all the languages in ODIN. To answer this question, we ran three experiments on the German data (for which there are 1757 IGT examples in ODIN after removing duplicates): (a) trained on the (gloss, translation) pairs for all IGT data, (b) trained on the (gloss, translation) pairs of the German data alone, and (c) trained on the (source, translation) pairs of the German data. The test was run against 58 IGT examples, a subset of the German test data in Table 1. It turns out that (a) performs much better than (b) and (c), which justifies the approach we proposed in Section 3.2. For instance, the F-measures for the *refined* alignment for (a)-(c) are 92.5%, 90.2%, and 85.6%, respectively.

[16]For the second setting, we wrote a 90-line Perl application that finds the root for each English word by using a dozen regular expression patterns combined with a list of 163 irregular verbs with their inflected forms.

Table 6: The performance of heuristic word aligner

	Precision	Recall	F-measure
No morphing	0.983	0.742	0.846
With morphing	0.983	0.854	0.914
Augmented aligner	0.981	0.881	**0.928**

4.3 Projection results

We evaluated the results of the major steps in our algorithm: the English DS derived from the parse trees produced by the English parser, the word alignment between the gloss and translation lines, and the projected source DS. We calculated the precision, recall, and F-score of the dependency links and word alignment links. The F-scores are shown in Table 7.[18]

Both the English parser and the word aligner work reasonably well with most F-scores well above 90%. The F-scores for dependency links in the source DS are lower partly due to errors in early parts of the process (e.g., English DS and word alignment), which propagates to this step. When we replace the automatically generated English DS and word alignment with the ones in gold standard, the F-measure of source DS increases significantly, as shown in Table 8.

To identify the causes of the remaining errors in the *oracle* results, we manually checked and classified one third of the errors in the German data. Among the 43 errors in the source DS, 26 (60.5%) are due to language divergence (e.g., head switching), eight (18.6%) are errors made by the projection heuristics, and nine (20.9%) are due to non-exact translations such as the one shown in Ex (2). Because language divergence can reveal interesting typological distinctions between languages, the first type of error may, in fact, identify examples that could be of great value to linguists and computational linguists.

(2)	der Antrag	des	oder	der	Dozenten
	the petition	of-the.SG	or	of-the.PL	docent.MSC

“the petition of the docent.” (Daniels, 2001)

5 Discussion

5.1 The IGT bias and knowledge discovery from enriched data

From the enriched data, various kinds of information can be extracted, such as grammars and transfer rules. We extracted CFGs for the seven languages by reading off the context-free rules from the projected

[17]We treat a word pair, (e,f), as a good translation if and only if both $P(e|f)$ and $P(f|e)$ are high.

[18]The *Total* word alignment F-measure is higher than 0.928 as mentioned in Table 6 because the test set used here is the superset of the one used in that section.

Table 7: The system performance on the seven languages

	GER	KKN	HUA	MEX	WLS	GLI	YAQ	Total
English DS	94.25	89.78	96.15	95.51	91.49	93.53	93.57	93.48
Word alignment	94.91	94.20	94.71	94.26	95.65	88.11	93.64	94.03
Source DS	78.14	82.16	84.71	84.22	84.39	78.17	79.36	81.45

Table 8: The F-measure of source dependency links with perfect English DS and/or word alignment

	GER	KKN	HUA	MEX	WLS	GLI	YAQ	Total
With gold Eng DS	82.21	87.67	88.46	85.23	91.72	80.16	83.81	85.42
With gold alignment	85.77	86.15	86.07	88.44	84.98	82.40	86.27	86.00
With both	91.21	91.67	89.82	89.65	94.25	85.77	90.68	90.64

Table 9: Extracted CFGs and evidence of word order

	HUA	MEX	GLI	YAQ
Word order	SVO	VOS	VSO	SOV
# of rule types	102	129	86	115
# of rule tokens	384	466	202	295

source PS. The numbers of rule types and rule tokens for four of the languages are listed in Table 9.

It is important to note that IGT data is somewhat biased: examples tend to be short and are selected for the purposes of a particular rhetorical context. They, therefore, deviate from the "normal" usage that one might normally expect to find in a corpus of language data. As such, one might question whether the information extracted from IGT would also be skewed due to these biases.

To test the usefulness of the data for answering typological questions, we wrote a tool that predicted the canonical word order (e.g., SOV, SVO) of a language using simple heuristics. It was able to produce the correct answers for all seven languages in our sample.[19] [20] We suspect that the number of IGT instances and their diversity (i.e., from multiple documents) is crucial to overcoming the IGT bias, and feel that the same heuristics could be applied to a much larger sample of languages. These could be further adapted to additional typological parameters beyond word order (e.g., orders of heads and modifiers in PS). We leave this to future work.

Given syntactically enriched data, it is also possible to search for patterns that are linguistically interesting. For instance, we wrote a piece of code that automatically identified examples with crossing dependencies (i.e., the ones whose DS have crossing links). One such example from the Yaqui data is in Ex (3), where the coordinated noun phrase *kow-ta into mis-ta* "the pig and the cat" is separated by the verb *bwuise-k* "grasp". Note that the crossing dependencies can only be discovered in the Yaqui data and not in the English since none exist in the English.

(3)	inepo	kow-ta	bwuise-k	into mis-ta
	1SG	pig-NNOM.SG	grasp-PST	and cat-NNOM.SG

"I caught the pig and the cat." (Martínez Fabián, 2006)

So far, we have examined linguistically interesting information in the source. In the future, we plan to examine structures in both the source and English. For instance, we plan to extract transfer rules from the aligned source and English structures and also calculate head/modifier crossings between languages similar to those described in (Fox, 2002).

5.2 Tools and resource building

The information that we discover about a language can help with the development of tools for the language. The order of constituents, for instance, can be used to inform prototype-driven learning strategies (Haghighi and Klein, 2006), which can then be applied to raw corpora. It is also possible that small samples of data showing the alignment interactions between source language structures and those of English can provide essential bootstrap information for informing machine translation systems (cf (Quirk and Corston-Oliver, 2006)).

Proof of the utility of an enriched corpus built over ODIN will depend crucially on its evaluation, and we feel that an important part of our future work will be the development of parsers that have been trained on projected structures. These parsers can be evaluated against human built corpora such as treebanks (obviously, only for those languages that have treebanks). Proof will also come from linguists who will be able to use the corpus to search for constructions of interest (*e.g.*, passives, relative clauses, etc.), and will likely be able to do so using standard tools such as

[19]Our code simply went through all the rules in the extracted CFGs and checked the position of the verb with respect to its subject and object. The -SBJ and -OBJ function tags were added to the English parse trees using simple heuristics and were carried over to the source PS via the projection algorithm.

[20]There is disagreement among linguists about German's underlying word order, being either SVO or SOV. Our heuristics returned SOV.

tgrep.[21] Crucially, linguists would be able to conduct such searches over a very large number of languages.

6 Conclusion

In this paper we demonstrate a methodology for projecting structure from annotated English data onto source language data. Because each IGT instance provides an English translation and an intermediary gloss line, we are able to project full syntactic structures from the automatically parsed translation. The fact that our basic methodology and code were applied to a typologically diverse sample of seven languages *without modification* suggests the potential for application to a much larger sample, perhaps numbering into the hundreds of languages. The resulting enriched structures could be of great importance to the fields of linguistics and computational linguistics. For the former, search facilities could be built over the data that would allow linguists to find syntactically marked up data for a large variety of languages, and could even accommodate cross-linguistic comparisons and analyses. For the latter, we could automatically discern grammars and transfer rules from the aligned and marked up data, where these computational artifacts could act as bootstraps for the development of additional tools and resources.

References

John Frederick Bailyn. 2001. Inversion, dislocation and optionality in russian. In Gerhild Zybatow, editor, *Current Issues in Formal Slavic Linguistics*.

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

Eugene Charniak. 1997. Statistical Parsing with a Context-Free Grammar and Word Statistics. In *Proc. of AAAI-1997*.

Michael W. Daniels. 2001. On a type-based analysis of feature neutrality and the coordination of unlikes. In *Proceedings of the 8th International HPSG Conference*. CSLI Publications.

Bonnie J. Dorr. 1994. Machine translation divergences: a formal description and proposed solution. *Computational Linguistics*, 20(4):597–635.

Heidi Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP 2002*, Philadelphia, Pennsylvania.

Raymond G. Gordon, editor. 2005. *Ethnologue: Languages of the World*. SIL International, Dallas, TX, fifteenth edition.

Aria Haghighi and Dan Klein. 2006. Protoype-driven sequence models. In *Proceedings of HLT-NAACL*, New York City, NY.

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*, Philadelphia, Pennsylvania.

Michael Krauss. 1992. The World's Languages in Crisis. *Language*, 68(1):4–10.

William D. Lewis. 2006. ODIN: A Model for Adapting and Enriching Legacy Infrastructure. In *Proceedings of the e-Humanities Workshop*, Amsterdam. Held in cooperation with e-Science 2006: 2nd IEEE International Conference on e-Science and Grid Computing.

David M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, Cambridge, Massachusetts, USA.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, et al. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proc of ARPA Speech and Natural Language Workshop*.

Constantino Martínez Fabián. 2006. *Yaqui Coordination*. Ph.D. thesis, University of Arizona.

Franz-Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *the 38th Annual Conference of the Association for Computational Linguistics (ACL-2000)*, pages 440–447.

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP 2006*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency tree translation: Syntactically informed phrasal smt. In *Proceedings of ACL 2005*.

John R. Swanton. 1912. Haida songs. In Franz Boas, editor, *Publications of the American Ethnological Society, Volume III*. E. J. Brill.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translation equivalence. In *Proceedings of ACL 2006*.

Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-English languages. In *Proceedings of HLT-EMNLP*, pages 851–858, Vancouver, British Columbia, Canada.

David Yarowksy and Grace Ngai. 2001. Inducing Multilingual POS taggers and NP Bracketers via robust projection across aligned corpora. In *Proceedings of NAACL-2001*, pages 377–404.

[21] This kind of search is reminiscent of Resnik's Linguists Search Engine (http://lse.umiacs.umd.edu), which allows structural search across text found on the Web.

Combining Lexical and Grammatical Features to Improve Readability Measures for First and Second Language Texts

Michael J. Heilman **Kevyn Collins-Thompson** **Jamie Callan** **Maxine Eskenazi**

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
4502 Newell Simon Hall
Pittsburgh, PA 15213-8213
{mheilman,kct,callan,max}@cs.cmu.edu

Abstract

This work evaluates a system that uses interpolated predictions of reading difficulty that are based on both vocabulary and grammatical features. The combined approach is compared to individual grammar- and language modeling-based approaches. While the vocabulary-based language modeling approach outperformed the grammar-based approach, grammar-based predictions can be combined using confidence scores with the vocabulary-based predictions to produce more accurate predictions of reading difficulty for both first and second language texts. The results also indicate that grammatical features may play a more important role in second language readability than in first language readability.

1 Introduction

The REAP tutoring system (Heilman, et al. 2006), aims to provide authentic reading materials of the appropriate difficulty level, in terms of both vocabulary and grammar, for English as a Second Language students. An automatic measure of readability that incorporated both lexical and grammatical features was thus needed.

For first language (L1) learners (i.e., children learning their native tongue), reading level has been predicted using a variety of techniques, based on models of a student's lexicon, grammatical surface features such as sentence length (Flesch, 1948), or combinations of such features (Schwarm and Ostendorf, 2005). It was shown by Collins-Thompson and Callan (2004) that a vocabulary-based language modeling approach was effective at predicting the readability of grades 1 to 12 of Web documents of varying length, even with high levels of noise.

Prior work on first language readability by Schwarm and Ostendorf (2005) incorporated grammatical surface features such as parse tree depth and average number of verb phrases. This work combining grammatical and lexical features was promising, but it was not clear to what extent the grammatical features improved predictions.

Also, discussions with L2 instructors suggest that a more detailed grammatical analysis of texts that examines features such as passive voice and various verb tenses can provide better features with which to predict reading difficulty. One goal of this work is to show that the use of pedagogically motivated grammatical features (e.g., passive voice, rather than the number of words per sentence) can improve readability measures based on lexical features alone.

One of the differences between L1 and L2 readability is the timeline and processes by which first and second languages are acquired. First language acquisition begins at infancy, and the primary grammatical structures of the target language are acquired by age four in typically developing children (Bates, 2003). That is, most grammar is acquired prior to the beginning of a child's formal education. Therefore, most grammatical features seen at high reading levels such as high school are present with similar frequencies at low reading levels such as grades 1-3 that correspond to elementary school-age children. It should be noted that sentence length is one grammar-related difference that can be observed as L1 reading level increases. Sentences are kept short in texts for low L1 reading levels in order to reduce the cognitive load on child readers. The average sentence length of texts increases with the age and reading level of the intended audience. This phenomenon has been utilized in early readability measures (Flesch, 1948). Vocabulary change, however, continues even into adulthood, and has been shown to be a more effective predictor of L1 readability than simpler measures such as sentence length (Collins-Thompson and Callan, 2005).

Second language learners, unlike their L1 counterparts, are still very much in the process of acquiring the grammar of their target language. In fact, even intermediate and advanced students of second languages, who correspond to higher L2 reading levels, often struggle with the grammatical structures of their target language. This phenomenon suggests that grammatical features may play a more important role in predicting and measuring L2 readability. That is not to say, however, that vocabulary cannot be used to predict L2 reading levels. Second language learners are learning both vocabulary and grammar concurrently, and reading materials for this population are chosen or authored according to both lexical and grammatical complexity. Therefore, the authors predict that a readability measure for texts intended for second language learners that incorporates both grammatical and lexical features could clearly outperform a measure based on only one of these two types of features.

This paper begins with descriptions of the language modeling and grammar-based prediction systems. A description of the experiments follows that covers both the evaluation metrics and corpora used. Experimental results are presented, followed by a discussion of these results, and a summary of the conclusions of this work.

2 Language Model Readability Prediction for First Language Texts

Statistical language modeling exploits patterns of use in language. To build a statistical model of text, training examples are used to collect statistics such as word frequency and order. Each training example has a label that tells the model the 'true' category of the example. In this approach, one statistical model is built for each grade level to be predicted.

The statistical language modeling approach has several advantages over traditional readability formulas, which are usually based on linear regression with two or three variables. First, a language modeling approach generally gives much better accuracy for Web documents and short passages (Collins-Thompson and Callan, 2004). Second, language modeling provides a probability distribution across *all* grade models, not just a single prediction. Third, language modeling provides more data on the relative difficulty of each word in the document. This might allow an application, for example, to provide more accurate vocabulary assistance.

The statistical model used for this study is based on a variation of the multinomial Naïve Bayes classifier. For a given text passage T, the semantic difficulty of T relative to a specific grade level G_i is predicted by calculating the likelihood that the words of T were generated from a representative language model of G_i. This likelihood is calculated for each of a number of language models, corresponding to reading difficulty levels. The reading difficulty of the passage is then estimated as the grade level of the language model most likely to have generated the passage T.

The language models employed in this work are simple: they are based on unigrams and assume that the probability of a token is independent of the surrounding tokens. A unigram language model is simply defined by a list of types (words) and their individual probabilities. Although this is a weak model, it can be effectively trained from less labeled data than more complex models, such as bigram or trigram models. Additionally, higher order n-gram models might capture grammatical as well as lexical differences. The relative contributions of grammatical and lexical features were thus better distinguished by using unigram language models that more exclusively focus on lexical differences.

In this language modeling approach, a generative model is assumed for a passage T, in which a hypothetical author generates the tokens of T by:

1. Choosing a grade language model, G_i, from the set $G = \{G_i\}$ of 12 unigram language models, according to a prior probability distribution P(G_i).

2. Choosing a passage length $|T|$ in tokens according to a probability distribution P($|T|$).

3. Sampling $|T|$ tokens from G_i's multinomial word distribution according to the 'naïve' assumption that each token is independent of all other tokens in the passage, given the language model G_i.

These assumptions lead to the following expression for the probability of T being generated by language model G_i according to a multinomial distribution:

$$P(T \mid G_i) = P(|T|)\,|T|!\prod_{w\in V}\frac{P(w \mid G_i)^{C(w)}}{C(w)!}$$

Next, according to Bayes' Theorem:

$$P(G_i \mid T) = \frac{P(G_i)P(T \mid G_i)}{P(T)}.$$

Substituting (1) into (2), taking logarithms, and simplifying produces:

$$\log P(G_i \mid T) = \sum_{w\in V} C(w)\log P(w \mid G_i) - \sum_{w\in V}\log C(w)! + \log R + \log S ,$$

where V is the list of all types in the passage T, w is a type in V, and C(w) is the number of tokens with type w in T. For simplicity, the factor R represents the contribution of the prior P(G_i), and S represents the contribution of the passage length $|T|$, given the grade level.

Two further assumptions are made to simplify the illustration:

1. That all grades are equally likely *a priori*. That is, $P(G_i) = \frac{1}{N_G}$ where N_G is the number of grade levels. For example, if there are 12 grade levels, then $N_G = 12$. This allows log R to be ignored.

2. That all passage lengths (up to a maximum length M) are equally likely. This allows log S to be ignored.

These may be poor assumptions in a real application, but they can be easily included or excluded in the model as desired. The log $C(w)!$ term can also be ignored because it is constant across levels. Under these conditions, an extremely simple form for the grade likelihood remains. In order to find which model G^i maximizes Equation (3), the model which G^i that maximizes the following equation must be found:

$$L(T \mid G_i) = \sum_{w\in V} C(w)\log P(w \mid G_i)$$

This is straightforward to compute: for each token in the passage T, the log probability of the token according to the language model of G^i is calculated. Summing the log probabilities of all tokens produces the overall likelihood of the passage, given the grade. The grade level with the maximum likelihood is then chosen as the final readability level prediction.

This study employs a slightly more sophisticated extension of this model, in which a sliding window is moved across the text, with a grade prediction being made for each window. This results in a distribution of grade predictions. The grade level corresponding to a given percentile of this distribution is chosen as the prediction for the entire document. The values used in these experiments for the percentile thresholds for L1 and L2 were chosen by accuracy on held-out data.

3 Grammatical Construction Readability Prediction for Second Language Texts

The following sections describe the approach to predicting readability based on grammatical features. As with any classifier, two components are required to classify texts by their reading level: first, a definition for and method of identifying features; second, an algorithm for using these features to classify a given text. A third component, training data, is also necessary in this classification task. The corpus of materials used for training and testing is discussed in a subsequent section.

3.1 Features for Grammar-based Prediction

L2 learners usually learn grammatical patterns explicitly from grammar explanations in L2 textbooks, unlike their L1 counterparts who learn them implicitly through natural interactions. Grammatical features would therefore seem to be an essential component of an automatic readability measure for L2 learners, who must actively acquire both the lexicon and grammar of their target language.

The grammar-based readability measure relies on being able to automatically identify grammatical constructions in text. Doing so is a multi-step process that begins by syntactically parsing the document. The Stanford Parser (Klein and Manning, 2002) was used to produce constituent structure trees. The choice of parser is not essential to the approach, although the accuracy of parsing does play a role in successful identification of certain grammatical patterns. PCFG scores from the parser were also used to filter out some of the ill-formed text present in the test corpora. The default training set of Penn Treebank (Marcus et al. 1993) was used for the parser because the domain and style of those texts actually matches fairly well with the domain and style of the texts on which a reading level predictor for second language learners might be used.

Once a document is parsed, the predictor uses Tgrep2 (Rohde, 2005), a tree structure searching tool, to identify instances of the target patterns. A Tgrep2 pattern defines dominance, sisterhood, precedence, and other relationships between nodes in the parse tree for a sentence. A pattern can also place constraints on the terminal symbols (e.g., words and punctuation), such that a pattern might require a form of the copula "be" to exist in a certain position in the construction. An example of a TGrep2 search pattern for the progressive verb tense is the following:

"VP < /^VB/ < (VP < VBG)"

Searching for this pattern returns sentences in which a verb phrase (VP) dominates an auxiliary verb (whose symbol begins with VB) as well as another verb phrase, which in turn dominates a verb in gerund form (VBG). An example of a matching sentence is, "The student was reading a book," shown in Figure 2.

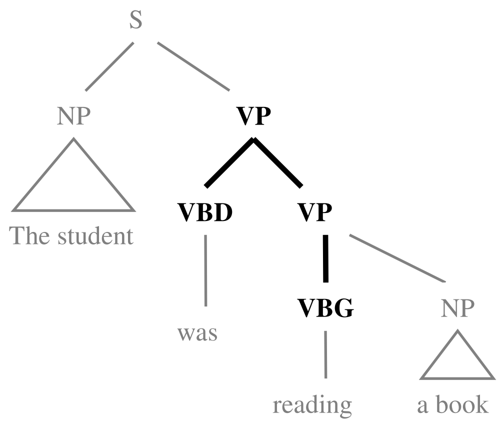


Figure 2: The parse tree for an example sentence that matches a pattern for progressive verb tense.

A set of 22 relevant grammatical constructions were identified from grammar textbooks for three different ESL levels (Fuchs et al., 2005). These grammar textbooks had different authors and publishers than the ones used in the evaluation corpora in order to minimize the chance of experimental results not generalizing beyond the specific materials employed in this study. The ESL levels correspond to the low-intermediate (hereafter, level 3), high-intermediate (level 4), and advanced (level 5) courses at the University of Pittsburgh's English Language Institute. The constructions identified in these grammar textbooks were then implemented in the form of Tgrep2 patterns.

Feature	Lowest Level	Highest Level
Passive Voice	0.11	0.71
Past Participle	0.28	1.63
Perfect Tense	0.01	0.33
Relative Clause	0.54	0.60
Continuous Tense	0.19	0.27
Modal	0.80	1.44

Table 1: The rates of occurrence per 100 words of a few of the features used by the grammar-based predictor. Rates are shown for the lowest (2) and highest (5) levels in the L2 corpus.

The rate of occurrence of constructions was calculated on a per word basis. A per-word rather than a per-sentence measure was chosen because a per-sentence measure would depend too greatly on sentence length, which also varies by level. It was also desirable to avoid having sentence length confounded with other features. Table 1 shows that the rates of occurrence of certain constructions become more frequent as level increases. This systematic variation across levels is the basis for the grammar-based readability predictions.

A second feature set was defined that consisted of 12 grammatical features that could easily be identified without computationally intensive syntactic parsing. These features included sentence length, the various verb forms in English, including the present, progressive, past, perfect, continuous tenses, as well as part of speech labels for words. The goal of using a second feature set was to examine how dependent prediction quality was on a specific set of features, as well as to test the extent to which the output of syntactic parsing might improve prediction accuracy.

3.2 Algorithm for Grammatical Feature-based Classification

A k-Nearest Neighbor (kNN) algorithm is used for classification based on the grammatical features described above. The kNN algorithm is an instance-based learning technique originally developed by Cover and Hart (1967) by which a test instance is classified according to the classifications of a given number (k) of training instances closest to it. Distance is defined in this work as the Euclidean distance of feature vectors. Mitchell (1997) provides more details on the kNN algorithm. This algorithm was chosen because it has been shown to be effective in text classification tasks when compared to other popular methods (Yang 1999). A k value of 12 was chosen because it provided the best performance on held-out data.

Additionally, it is straightforward to calculate a confidence measure with which kNN predictions can be combined with predictions from other classifiers—in this case with predictions from the unigram language modeling-based approach described above. A confidence measure was important in this task because it provided a means with which to combine the grammar-based predictions with the predictions from the language modeling-based predictor while maintaining separate models for each type of feature. These separate models were maintained to better determine the relative contributions of grammatical and lexical features.

A static linear interpolation of predictions using the two approaches led to only minimal reductions of prediction error, likely because predictions from the poorer performing grammar-based classifier were always given the same weight. However, with the confidence measures, predictions from the grammar-based classifier could be given more weight when the confidence measure was high, and less weight when the measure was low and the predictions were likely to be inaccurate. The case-dependent interpolation of prediction values allowed for the effective combination of language modeling- and grammar-based predictions.

The confidence measure employed is the proportion of the *k* most similar training examples, or nearest neighbors, that agree with the final label chosen for a given test document. For example, if seven of ten neighbors have the same label, then the confidence score will be 0.6. The interpolated readability prediction value is calculated as follows:

$$L_I = L_{LM} + C_{kNN} * L_{GR},$$

where L_{LM} is the language model-based prediction, L_{GR} is the grammar-based prediction from the kNN algorithm, and C_{kNN} is the confidence value for the kNN prediction. The language modeling approach is treated as a black box, but it would likely be beneficial to have confidence measures for it as well.

4 Descriptions of Experiments

This section describes the experiments used to test the hypothesis that grammar-based features can improve readability measures for English, especially for second language texts. The measures and cross-validation setup are described. A description of the evaluation corpora of labeled first and second language texts follows.

4.1 Experimental Setup

Two measurements were used in evaluating the effectiveness of the reading level predictions. First, the correlation coefficient evaluated whether the trends of prediction values matched the trends for human-labeled texts. Second, the mean squared error of prediction values provided a measure of how correct each of the predictors was on average, penalizing more severe errors more heavily. Mean square error was used rather than simple accuracy (i.e., number correct divided by sample size) because the task of readability prediction is more akin to regression than classification. Evaluation measures such as accuracy, precision, and recall are thus less meaningful for readability prediction tasks because they do not capture the fact that an error of 4 levels is more costly than an error of a single level.

A nine-fold cross-validation was employed. The data was first split into ten sets. One set was used as held-out data for selecting the parameter *k* for the kNN algorithm and the percentile value for the language modeling predictor, and then the remaining nine were used to evaluate the quality of predictions. Each of these nine was in turn selected as the test set, and the other eight were used as training data.

4.2 Corpora of Labeled Texts

Two corpora of labeled texts were used in the evaluation. The first corpus was from a set of texts gathered from the Web for a prior evaluation of the language modeling approach. The 362 texts had been assigned L1 levels (1-12) by grade school teachers, and consisted of approximately 250,000 words. For more details on the L1 corpus, see (Collins-Thompson and Callan, 2005).

The second corpora consisted of textbook materials (Adelson-Goldstein and Howard, 2004, for level 2; Ediger and Pavlik, 2000, for levels 3 and 4; Silberstein, 2002, for level 5) from a series of English as a Second Language reading courses at the English Language Institute at the University of Pittsburgh. The four reading practice textbooks that constitute this corpus were from separate authors and publishers than the grammar textbooks used to select and define grammatical features. The reading textbooks in the corpus are used in courses intended for beginning (level 2) through advanced (level 5) students. The textbooks were scanned into electronic format, and divided into fifty roughly equally sized files. This second language corpus consisted of approximately 200,000 words.

Although the sources and formats of the two corpora were different, they share a number of characteristics. Their size was roughly equal. The documents in both were also fairly but not perfectly evenly distributed across the levels. Both corpora also contained a significant amount of noise which made accurate prediction of reading level more challenging. The L1 corpus was from the Web, and therefore contained navigation menus, links, and the like. The texts in the L2 corpus also contained significant levels of noise due to the inclusion of directions preceding readings, exercises and questions following readings, as well as labels on figures and charts. The scanned files were not hand-corrected in this study, in part to test that the measures are robust to noise, which is present in the Web documents for which the readability measures are employed in the REAP tutoring system.

The grammar-based prediction seems to be more significantly negatively affected by the noise in the two corpora because the features rely more on dependencies between different words in the text. For example, if a word happened to be part of an image caption rather than a well-formed sentence, the unigram language modeling approach would only be affected for that word, but the grammar-based approach might be affected for features spanning an entire clause or sentence.

5 Results of Experiments

The results show that for both the first and second language corpora, the language modeling (LM) approach alone produced more accurate predictions than the grammar-based approach alone. The mean squared error values (Table 2) were lower, and the correlation coefficients (Table 3) were higher for the LM predictor than the grammar-based predictor.

The results also indicate that while grammar-based predictions are not as accurate as the vocabulary-based scores, they can be combined with vocabulary-based scores to produce more accurate interpolated scores. The interpolated predictions combined by using the kNN confidence measure were slightly and in most tests significantly more accurate in terms of mean squared error than the predictions from either single measure. Interpolation using the first set of grammatical features led to 7% and 22% reductions in mean squared error on the L1 and L2 corpora, respectively. These results were verified using a one-tailed paired t-test of the squared error values of the predictions, and significance levels are indicated in Table 2.

Mean Squared Error Values		
Test Set (Num. Levels)	**L1(12)**	**L2(4)**
Language Modeling	5.02	0.51
Grammar	10.27	1.08
Interpolation	4.65*	0.40**
Grammar2 (feature set #2)	12.77	1.26
Interp2. (feature set #2)	4.73	0.43*

Table 2. Comparison of Mean Squared Error of predictions compared to human labels for different methods. Interpolated values are significantly better compared to language modeling predictions where indicated (* = $p<0.05$, ** = $p<0.01$).

Correlation Coefficients		
Test Set (Num. Levels)	**L1(12)**	**L2(4)**
Language Modeling	0.71	0.80
Grammar	0.46	0.55
Interpolation	0.72	0.83
Grammar2 (feature set #2)	0.34	0.48
Interp2. (feature set #2)	0.72	0.81

Table 3. Comparison of Correlation Coefficients of prediction values to human labels for different prediction methods.

The trends were similar for both sets of grammatical features. However, the first set of features that included complex syntactic constructs led to better performance than the second set, which included only verb tenses, part of speech labels, and sentence length. Therefore, when syntactic parsing is not feasible because of corpora size, it seems that grammatical features requiring only part-of-speech tagging and word counts may still improve readability predictions. This is practically important because parsing can be too computationally intensive for large corpora.

All prediction methods performed better, in terms of correlations, on the L2 corpus than on the L1 corpus. The L2 corpus is somewhat smaller in size and should, if only on the basis of training material available to the prediction algorithms, actually be more difficult to predict than the L1 corpus. To ensure that the range of levels was not causing the four-level L2 corpus to have higher predictions than the twelve-level L1 corpus, the L1 corpus was also divided into four bins (grades 1-3, 4-6, 7-9, 10-12). The accuracy of predictions for the binned version of the L1 corpus was not substantially different than for the 12-level version.

6 Discussion

In the experimental tests, the LM approach was more effective for measuring both L1 and L2 readability. There are several potential causes of this effect. First, the language modeling approach can utilize all the words as they appear in the text as features, while the grammatical features were chosen and defined manually. As a result, the LM approach can make measurements on a text for as many features as there are words in its lexicon. Additionally, the noise present in the corpora likely affected the grammar-based approach disproportionately more because that method relies on accurate parsing of relationships between words.

Additionally, English is a morphologically impoverished language compared to most languages. Text classification, information retrieval, and many other human language technology tasks can be accomplished for English without accounting for grammatical features such as morphological inflections. For example, an information retrieval system can perform reasonably well in English without performing stemming, which does not greatly increase performance except when queries and documents are short (Krovetz, 1993).

However, most languages have a rich morphology by which a single root form may have thousands or perhaps millions of inflected or derived forms. Language technologies must account for morphological features in such languages or the vocabulary grows so large that it becomes unmanageable. Lee (2004), for example, showed that morphological analysis can improve the quality of statistical machine translation for Arabic. Thus it seems that grammatical features could contribute even more to measures of readability for texts in other languages.

That said, the use of grammatical features appears to play a more important role in readability measures for L2 than for L1. When interpolated with grammar-based scores, the reduction of mean squared error over the language modeling approach for L1 was only 7%, while for L2 the reduction or squared error was 22%. An evaluation on corpora with less noise would likely bring out these differences further and show grammar to be an even more important factor in second language readability. This result is consistent with the fact that second language learners are still in the process of acquiring the basic grammatical constructs of their target language.

7 Conclusion

The results of this work suggest that grammatical features can play a role in predicting reading difficulty levels for both first and second language texts in English. Although a vocabulary-based language modeling approach outperformed the grammar-based predictor, an interpolated measure using confidence scores for the grammar-based predictions showed improvement over both individual measures. Also, grammar appears to play a more important role in second language readability than in first language readability. Ongoing work aims to improve grammar-based readability by reducing noise in training data, automatically creating larger grammar feature sets, and applying more sophisticated modeling techniques.

8 Acknowledgements

We would like to acknowledge Lori Levin for useful advice regarding grammatical constructions, as well as the anonymous reviewers for their suggestions.

This material is based on work supported by NSF grant IIS-0096139 and Dept. of Education grant R305G03123. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsors.

References

J. Adelson-Goldstein and L. Howard. 2004. *Read and Reflect 1*. Oxford University Press, USA.

E. Bates. 2003. On the nature and nurture of language. In R. Levi-Montalcini, D. Baltimore, R. Dulbecco, F. Jacob, E. Bizzi, P. Calissano, & V. Volterra (Eds.), *Frontiers of biology: The brain of Homo sapiens* (pp. 241–265). Rome: Istituto della Enciclopedia Italiana fondata da Giovanni Trecanni.

M. Fuchs, M. Bonner, M. Westheimer. 2005. *Focus on Grammar,* 3rd Edition. Pearson ESL.

K. Collins-Thompson and J. Callan. 2004. A language modeling approach to predicting reading difficulty. *Proceedings of the HLT/NAACL Annual Conference*.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13, 21-27.

A. Ediger and C. Pavlik. 2000. *Reading Connections Intermediate*. Oxford University Press, USA.

A. Ediger and C. Pavlik. 2000. *Reading Connections High Intermediate*. Oxford University Press, USA.

M. Heilman, K. Collins-Thompson, J. Callan & M. Eskenazi. 2006. Classroom success of an Intelligent Tutoring System for lexical practice and reading comprehension. *Proceedings of the Ninth International Conference on Spoken Language Processing.*

D. Klein and C. D. Manning. 2002. Fast Exact Inference with a Factored Model for Natural Language Parsing. *Advances in Neural Information Processing Systems* 15 (NIPS 2002), December 2002.

R. Krovetz. 1993. Viewing morphology as an inference process. *SIGIR-93*, 191–202.

Y. Lee. 2004. Morphological Analysis for Statistical Machine Translation. *Proceedings of the HLT/NAACL Annual Conference*.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. "Building a large annotated corpus of English: the Penn Treebank." *Computational Linguistics*, 19(2).

T. Mitchell. 1997. *Machine Learning*. The McGraw-Hill Companies, Inc. pp. 231-236.

D. Rohde. 2005. *Tgrep2 User Manual*. http://tedlab.mit.edu/~dr/Tgrep2/tgrep2.pdf.

S. Schwarm, and M. Ostendorf. 2005. Reading Level Assessment Using Support Vector Machines and Statistical Language Models. *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

S. Silberstein, B. K. Dobson, and M. A. Clarke. 2002. *Reader's Choice,* 4th edition. University of Michigan Press/ESL.

Y. Yang. 1999. A re-examination of text categorization methods. *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR'99, pp 42--49).

Automatic Assessment of Student Translations for Foreign Language Tutoring

Chao Wang and Stephanie Seneff
Spoken Language Systems Group
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center, 32 Vassar Street, Cambridge, MA 02139
{wangc,seneff}@csail.mit.edu

Abstract

This paper introduces the use of speech translation technology for a new type of voice-interactive Computer Aided Language Learning (CALL) application. We describe a computer game we have developed, in which the system presents sentences in a student's native language to elicit spoken translations in the target new language. A critical technology is an algorithm to automatically verify the appropriateness of the student's translation using linguistic analysis. Evaluation results are presented on the system's ability to match human judgment of the correctness of a student's translation, for a set of 1115 utterances collected from 9 learners of Mandarin Chinese translating flight domain sentences. We also demonstrate the effective use of context information to improve both recognition performance on non-native speech as well as the system's accuracy in judging the translation quality.

1 Introduction

It is widely recognized that one of the best ways to learn a foreign language is through spoken dialogue with native speakers (Ehsani and Knodt, 1998). However, this is not a practical method in the classroom setting. A potential solution to this problem is to rely on computer spoken dialogue systems to role play a tutor and/or a conversational partner. Ideally, a voice-interactive system can provide the learner with endless opportunities for practice and feedback. However, while a number of dialogue systems have been developed (or adapted) for language learning purposes (Seneff et al., 2004; Johnson et al., 2004), the issues of speech understanding of the accented and disfluent utterances of a foreign language student typically lead to unacceptable performance (Eskenazi, 1999).

A relatively successful application of speech processing technology is in the area of pronunciation training (Eskenazi, 1999; Witt, 1999; Hardison, 2004). In this case, a learner repeats words or sentences prompted by the computer, and receives feedback on the segmental and suprasegmental quality of their speech. While such systems can be useful, they do not help the student learn how to formulate sentences within the language on their own.

In this paper, we introduce a novel speech translation game which aims to provide a fun and reliable environment for language learners to gain proficiency in communication. Our application is complementary to pronunciation training in that we provide the opportunity for the students to practice expressing themselves in the new language. While we don't explicitly evaluate their pronunciations, we provide implicit feedback in that they must speak well enough for the speech recognizer to successfully process their utterances. The translation exercise can also serve as a preparation stage in which students can practice relevant vocabulary and sentence constructs, in order to prepare themselves for the more challenging task of interacting with a dialogue system in the same domain.

System:	(*greet and prompt user*) Welcome! You are playing at level four. Okay, here's your first sentence to translate: **The return date is Tuesday October 25**.
User:	(*translate with error*) fan3 hui2 ri4 qi1 shi4 si4 yue4 er4 shi2 wu3 hao4 xing1 qi1 er4.
System:	(*paraphrase in Chinese*) fan3 hui2 ri4 qi1 si4 yue4 er4 shi2 wu3 hao4 xing1 qi1 er4.
System:	(*paraphrase in English*) The return date is Tuesday April 25.
User:	(*repeat date*) shi2 yue4 er4 shi2 wu3 hao4.
System:	(*paraphrase in Chinese*) shi2 yue4 er4 shi2 wu3 hao4.
System:	(*paraphrase in English*) October 25.
System:	(*respond to user*) "Great! You successfully translated the sentence in separate fragments. Let's move on to the next sentence: **I need to fly to Hong Kong the day after tomorrow**.
User:	(*click on the help button*)
System:	(*repeat English prompt*) I need to fly to Hong Kong the day after tomorrow.
System:	(*offer example translation*) wo3 xu1 yao4 hou4 tian1 fei1 xiang1 gang3.
User:	(*imitate the translation*) wo3 xu1 yao4 hou4 tian1 fei1 xiang1 gang3.
...	...
System:	(*respond to user*) You translated nine out of ten sentences. You took on average 1.6 turns per sentence. You have advanced to level five. Would you like to play another round?

Figure 1: Example interaction between a user and the system.

Our prototype centers on the task of translating phrases and sentences from English into Chinese, in the flight reservation domain. As illustrated by the example dialogue in Figure 1, the system role plays a language tutor interacting with a Mandarin learner. The system prompts the student with randomly generated English sentences to elicit spoken Chinese translations from the learner. The system paraphrases each user utterance in both languages, to implicitly inform the user of the system's internal understanding, and judges whether the student has succeeded in the task. The system keeps track of how many turns a user takes to complete all the sentences in a game session, and rewards good performance by advancing the student towards higher difficulty levels. A convenient "help" button allows the student to request a translation of the current game sentence, to help them overcome gaps in their knowledge of the linguistic constructs or the vocabulary. The student can also type any English sentences within the domain to obtain a reference translation. The system utilizes an interlingua-based bidirectional translation capability, described in detail in (Wang and Seneff, 2006; Seneff et al., 2006). Both Chinese and English sentences are parsed into a common meaning representation, which we loosely refer to as an "interlingua," from which paraphrases in both languages can be automatically generated using formal generation rules.

The key to a successful tutoring system lies in its ability to provide immediate and pertinent feedback on the student's performance, similar to a human tutor. A central focus of this paper is to address the challenging problem of automatically assessing the appropriateness of a student's translation. At first glance, our task appears to share much in common with machine translation (MT) evaluation (Hovy et al., 2002). Indeed, both are trying to assess the quality of the translation output, whether it is produced by a computer or by a foreign language student. Nevertheless, there also exist several fundamental distinctions. Automatic MT evaluation methods, as represented by the well-known Bleu metric (Papineni et al., 2001), assume the availability of human reference translations. The algorithms typically compare MT outputs with reference translations with the goal of producing a quality indicator (on a numeric scale) that correlates with human judgement. In contrast, our algorithm operates in the absence of human generated reference translations[1]. Furthermore, our application requires the evaluation algorithm to make accept/reject decisions on each *individual* translation, in the same way as a language tutor determines whether a translation is acceptable or not. While our task is more demanding, it is made possible by operating in restricted domains.

The remainder of the paper is organized as follows. In Section 2, we present an interlingua-based approach for verifying the correctness of the student's spoken translation. Section 3 describes the

[1]We employ a grammar of recursive rewrite rules to generate a very large number of English prompt sentences. It would be too costly and time-consuming to generate human translations to cover this space.

evaluation framework, followed by results and discussions in Section 4. Finally, we discuss future plans for extending our work.

2 Methodology

The two most important aspects in the human evaluation of translation quality are *fluency* and *fidelity* (Hovy et al., 2002). In our case, we consider a student's translation to be acceptable if it is well-formed (high fluency) and conveys the same meaning as the input sentence (high fidelity). We designed our interlingua-based evaluation algorithm following these two principles. The algorithm uses parsability to verify fluency. Fidelity is examined by extracting and comparing semantic information from the translation pairs. In the following, we begin by describing the basic steps involved in our translation verification algorithm. We then discuss different strategies for integrating with the speech recognition system.

2.1 Parsing

Our framework depends strongly on an ability to parse both the English and Chinese sentences into a common interlingual meaning representation. Parsing is critical both for producing the two paraphrases of the student's utterance and for judging the quality of their provided translation. Both English and Chinese grammars are needed to analyze the source and target sides of each translation pair. The grammars have been carefully constructed so that meaning representations derived from both languages are as similar as feasible.

We utilized a parser (Seneff, 1992) that is based on an enhanced probabilistic context-free grammar (PCFG), which captures dependencies beyond context-free rules by conditioning on the external left-context parse categories when predicting the first child of each parent node. While we use a specific grammar for analyzing flight domain sentences, we emphasize domain portability of the grammar by using mainly syntactic information in the majority of the parse tree rules. Semantics are introduced near the terminals, mostly involving adjectives, verbs, nouns and proper noun classes. Rules for general semantic concepts such as dates and times are organized into sub-grammars that are easily embedded into any domain. We have successfully applied the same strategy in developing both the Chinese and English grammars. Once a parse tree is obtained, selected parse categories are extracted to form a hierarchical meaning representation encoding both syntactic and semantic information.

2.2 Semantic Information Comparison

In principle, we can directly compare the meaning representations derived from the source and target sides of the translation pair to determine their equivalence. In practice, the meaning representation still captures too much language-specific detail, which makes the comparison prone to failure. Even the pair of English utterances, "How much is the second flight?" and "What is the price of the second flight?" have essentially the same meaning, but would not produce identical meaning representations. Across languages, this situation becomes much worse.

We adopted two complementary strategies to increase the chance of a match between the English prompt and the student translation. First, the English prompt is translated into a reference Chinese translation using the existing interlingua translation capability. This extra step aims at reducing discrepancies caused by syntactic structure differences between the two languages. Secondly, we abstract from the original meaning representation into a simple encoding of key-value (KV) pairs. This is accomplished using a language generation system (Baptist and Seneff, 2000), with generation rules determining what information to extract from the original hierarchical meaning representation. Figure 2 shows a couple of examples of the KV representation that we used for scoring.

Another important role of the KV generation step is to bring in a flexible mechanism for defining equivalence, which is a tricky task even for human evaluators. For example, while it is somewhat obvious that "(1) Give me flights leaving around nine p m" is equivalent to "(2) Give me flights departing around nine p m," it is unclear whether these two sentences are equivalent to "(3) Give me flights around nine p m" or even "(4) I would like to leave around nine p m." From a pragmatic point of view, the same speaker intention can be inferred from the four sentences. On the other hand, it can be argued that (1) and (2) are completely interchangeable

```
{c eform
   :topic  "fare"
   :airline_name "united"
   :trace "how much" }
```

lian2 he2 hang2 kong1 de5 piao4 jia4 shi4 duo1 shao3?
(*What is the fare on United airlines?*)

```
{c eform
   :topic "flight"
   :source "paris"
   :departure_time {c eform
                      :clock_hour 9
                      :xm "a m" }}
```

shang4 wu3 jiu3 dian3 cong2 ba1 li2 chu1 fa1 de5 ban1 ji1.
(*Flights from Paris leaving at nine o'clock in the morning.*)

Figure 2: Frame representation of the key-value information for two example Chinese sentences.

while (3) and (4) could not substitute for (1) or (2) in some circumstances. Criteria for equivalence can be controlled by what is extracted from the meaning representation. If only a `departure_time` key is generated for the sentences, then all four sentences will be equivalent. However, if more information is preserved in the KV pairs, for example, a `topic` key with value `flight`, then sentence (4) will not be considered as equivalent to sentences (1)-(3). Considering that our intended application is language tutoring, we lean towards a stricter criterion for defining equivalence. The KV generation rules are developed manually, guided by human-rated development data. The KV inventory includes over 80 unique keys.

Once the KV pairs are obtained from the prompt (reference) and the student translation (hypothesis), a recursive procedure is applied to compare all the keys in the reference and hypothesis KV frames. Mismatches are tabulated into substitutions (different values for the same key), deletions (extra keys in the reference), and insertions (extra keys in the hypothesis). A perfect match is achieved if there are no mismatch errors. Figure 3 summarizes the procedure to evaluate students' spoken translations.

Partial match for a good student translation is a common problem caused by speech recognition errors, particularly on dates and times. It is natural for the student to just repeat the "incorrect" piece after noticing the error in the system's paraphrases. Hence, in the tutoring application, we added a submatch mode to the comparison algorithm, which works in a divide-and-conquer manner. All matching KV pairs in each turn are checked off from the reference, and a subsequent submatch succeeds once there are no remaining KV pairs unaccounted for. One limitation of the incremental comparison algorithm is that it ignores insertion errors. The tutoring system provides a special reply message when a sentence is translated via partial matches accomplished over a series of utterances, to distinguish from the case of a perfect match in a single turn, as illustrated in the example dialogue.

2.3 Integration with Speech Recognition

A user's utterance is first processed by the speech recognizer to produce word hypotheses. The recognizer is configured from a segment-based speech recognition system (Glass, 2003), using Chinese acoustic models trained on native speakers' data (Wang et al., 2000a; Wang et al., 2000b). Tone features are ignored in the acoustic models; however, the language model implicitly captures some tone constraints. This is preferred over modeling tone explicitly, considering that non-native speakers typically make many tone errors. The language model was initially trained on Chinese translations of English sentences generated from the templates used in the game, and later augmented with additional data collected from users. The recognizer can output multiple hypotheses in the form of an *N*-best list. The parser is able to convert the *N*-best list into a lattice, and re-select a best hypothesis based on a combination of recognition and parsing scores.

Poor recognition on non-native speech is a major performance issue for CALL application. In our domain, dates, times, and flight numbers are particularly challenging entities for the recognizer. Recognition error typically results in false rejection, causing frustration to the user. Since the system has explicit knowledge of the sentence the student is trying to produce, it should be feasible to exploit this knowledge to improve speech understanding. A plausible strategy is to dynamically adjust the recognizer's language model in anticipation of what the user is likely to say, as exemplified by dialogue context dependent language models (Solsona et al., 2002).

In theory, we could use the automatically gener-

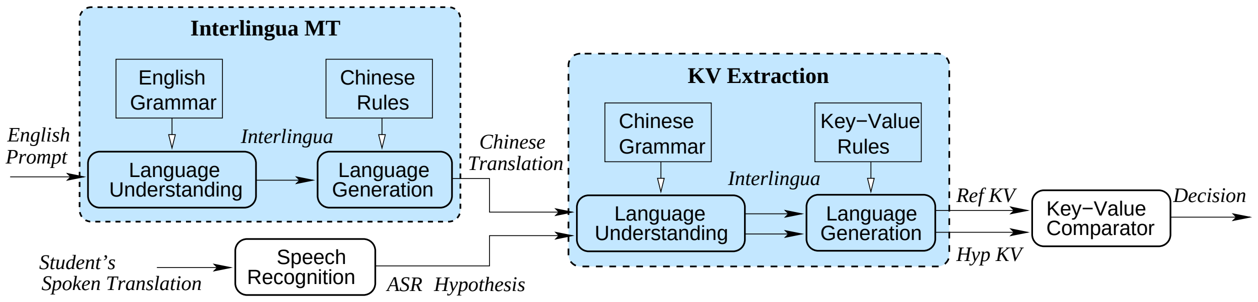


Figure 3: Schematic of procedure to evaluate students' spoken translations.

ated reference translation to explicitly bias the language model. However, one has to take care not to bias towards the correct response so strongly that the student is allowed to make mistakes with impunity. Furthermore, this strategy would not generalize to cover all the possible legitimate translations a student might produce for that prompt. Instead, we devised a simple strategy that overcomes these issues. We select a preferred hypothesis from the *N*-best list if its KV representation matches the reference. Thus the student has to speak well enough for a correct answer to appear somewhere in the N-best list, without any manipulations of the recognizer's language model. If the parser fails to find a perfect match in the *N*-best list, it will choose the hypothesis with the best score, or fall back to the recognizer's top hypothesis if no parse theory could succeed.

3 Evaluation Framework

Given a translation pair, the goal of our algorithm is to make the same accept/reject decision as a human evaluator. Hence, we can evaluate our algorithm in a classification framework. In this section, we first present the data collection and labeling effort. We then describe a baseline system based on a variant of the Bleu metric. Finally we briefly describe the metrics we used to evaluate our algorithms.

3.1 Data Collection and Labeling

During the course of developing a prototype game system, two developers and two student testers interacted extensively with the system. A total of 2527 Chinese waveforms, recorded during this process, became development data for finding gaps in the interlingua-based matching method and for tuning parameters for the baseline method.

For evaluation, we use 1115 utterances collected from 9 users with varying degrees of Chinese exposure. These subjects were asked to play the translation game over the Web and fill out a survey afterwards. They came from a rich background of Chinese exposure, include advanced "heritage" speakers of Chinese (including dialects such as Cantonese and Shanghainese), as well as novices who just completed two semesters of a college-level Chinese class.

The speech waveforms recorded from the interactions were manually transcribed with orthography, gender, and speaker information. The transcriber was instructed to transcribe spontaneous speech effects, such as false starts and filled pauses. However, tonal mispronunciations are completely ignored, and segmental errors are largely ignored to the extent that they do not result in a different syllable.

The translation pairs (the English prompt and the orthographic transcription of the student translation) were rated independently by two bilingual speakers to provide reference labels for evaluating the verification algorithm. The two raters, both native in Chinese and fluent in English, labelled each translation with either an "accept" or a "reject" label. Translations can be rejected because of bad language usage (including false starts) or because of mismatches in meaning. One labeller rated both development and test data, while the second labeller only rated the test data. The interlabeller agreement on the test data has a kappa score (Uebersax, 1998) of 0.85. The subset of data for which there was disagreement were relabelled by the two raters jointly to reach a consensus.

3.2 Baseline

The Bleu metric has been widely accepted as an effective means to automatically evaluate the quality of machine translation outputs (Papineni et al., 2001). An interesting question is whether it would be useful for the purpose of assessing the appropriateness of translations produced by *non-native speakers* at a sentence by sentence granularity level. We developed a simple baseline algorithm using the NIST score, which is a slight variation of Bleu[2]. Given an English prompt, the interlingua-based machine translation system first produces a reference translation. The student's translation is then compared against the machine output to obtain a NIST score. The translation is accepted if the score exceeds a certain threshold optimized on the development data.

Figure 4 plots the Receiver Operating Characteristics (ROC) curve of the baseline algorithm, obtained by varying the NIST score acceptance threshold. Each point on the curve represents a tradeoff between accepting an erroneous translation (False Accept) and rejecting a good one (False Reject). As shown in the plot, the NIST score based ROC curve is far from reaching the ideal top-left corner. For language tutoring purposes, it is desirable to operate in the low false acceptance region. However, a 20% false acceptance rate will result in the system rejecting over 35% of correct student translations. The operating point that minimizes overall classification error turns out to be biased towards leniency, falsely accepting over 60% of translations that are rejected by human raters. The resulting minimum error rate on development data transcripts is 23.0%, with a NIST score threshold of 3.16. The threshold for automatic speech recognition (ASR) outputs was optimized separately using the 1-best hypotheses of utterances in the development data. The optimal threshold on ASR outputs is 1.60, resulting in a classification error rate of 24.1%. The majority classifier, corresponding to the $(1, 1)$ point on the curve, translates into a 31.6% error rate on the development data.

3.3 Evaluation Metrics

We evaluated the overall system performance on test data using human decisions as ground truth. Although we can not generate an ROC curve for our proposed algorithm (because it is a non-parametric method), we plot its performance along with the ROC curve of the baseline system for a more thorough comparison.

[2]We determined empirically that the NIST score works slightly better than the Bleu score in our application. The scores are computed using the NIST MT scoring tool from: `ftp://jaguar.ncsl.nist.gov/mt /resources/mteval-v11b.pl`

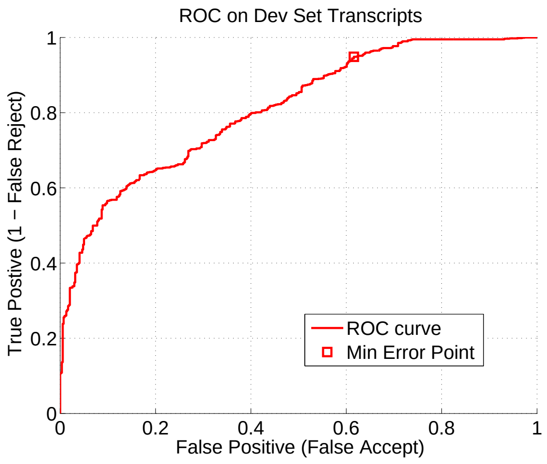


Figure 4: ROC curve by changing acceptance threshold on the NIST score on transcriptions of development data.

We evaluated the different ASR integration strategies (1-best hypothesis, 10-best hypotheses, using contextual constraints from reference KV) based on sentence classification error rates as well as speech recognition performance.

4 Results and Discussions

Table 1 summarizes the false accept, false reject, and overall classification error rates on unseen test data. With manual transcripts as inputs, the baseline algorithm using the NIST score achieved a classification error rate of 19.3%, as compared with 25.0% for the trivial case of always accepting the user sentence (Majority classifier). The KV-based algorithm achieved a much better performance, with only a 7.1% classification error rate. This translates into a kappa score of 0.86, which is slightly above the level of agreement initially achieved by the two labellers. Note that the performance difference compared to the baseline system is mostly attributed to a large reduction in the "False Accept" category.

Interestingly, the NIST method degrades only slightly when it is applied to the speech recognition 1-best output rather than the transcript. However, this result is deceptive, as it is now even more bi-

Transcript	False Reject	False Accept	Classification Error
Majority	0.0%	100%	25.0%
NIST	8.0%	54.5%	19.6%
KV	7.3%	6.8%	7.2%

ASR	False Reject	False Accept	Classification Error
NIST	4.2%	77.1%	22.4%
KV 1-best	32.1%	4.3%	25.1%
KV 10-best	27.0%	7.2%	22.1%
KV Context	13.5%	14.7%	13.8%

Table 1: Classification results for various evaluation systems, on both transcripts and automatic speech recognition (ASR) outputs. Note that the "KV Context" condition favors a hypothesis that matches the prompt KV.

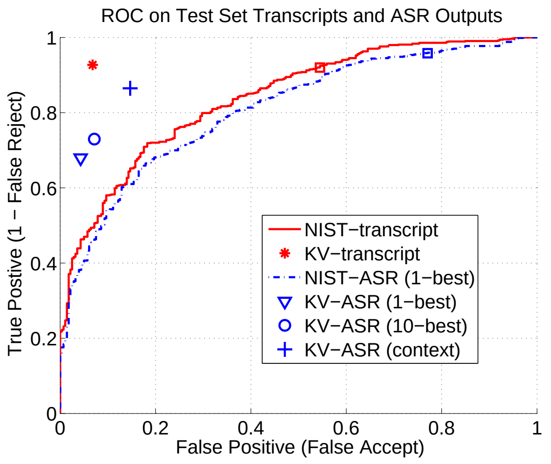


Figure 5: Comparison of ROC of different methods.

	Syllable		Sentence	
	ER(%)	RR(%)	ER(%)	RR(%)
1-best	11.6	-	40.4	-
10-best	10.7	7.8	38.7	4.2
Context	8.7	25.0	30.0	25.7

Table 2: Comparison of speech recognition performance in syllable error rates and sentence error rates, for three different strategies of utterance selection from an N-best list. (ER stands for error rate, RR stands for relative reduction.)

ased towards a "False Accept" strategy, causing over three quarters of the students' erroneous utterances to be accepted. The KV method is much more susceptible to speech recognition error because of its deep linguistic analysis. For instance, any recognition errors causing a parse failure will result in a "reject" decision, which explains the high error rate when only the 1-best hypothesis is used. However, the KV algorithm can improve substantially by searching the full N-best list ($N = 10$) for a plausible analysis. When contextual information (KV Context) is used, our simple strategy of favoring the hypothesis matching the reference KV reduces the classification error rate dramatically.

A plot of the receiver operating characteristics of these methods in Figure 5 reveals a clear picture of the performance differences. All of the KV points are clustered in the upper left corner of the plot, above the ROC curve of the NIST-based method. The NIST-score based classifier (represented by the square marker on the ROC curve) is heavily biased towards making the acceptance decision (the majority class). In contrast, the KV method operates in the low "False Accept" area. It achieves a much lower false rejection rate when compared with the NIST method operating at an equivalent false acceptance point.

Although the classification error rate clearly improves when the KV method makes use of the N-best list and incorporates contextual constraints, the ROC plot seems to suggest that the error reduction might simply be attributed to a shift in the operating point: the improvements are caused by a bias towards making the majority class decision. We use improvements in speech recognition to demonstrate that this is not the case (at least not entirely). Table 2 summarizes the syllable and sentence error rates on the test data, for the three configurations discussed previously (1-best, 10-best, and Context). By using a tighter integration with the parser with contextual constraints, we greatly improved speech recognition performance, marked by reductions of syllable and sentence error rates by 25% and 25.7% respectively.

5 Conclusions and Future Work

In this paper, we have presented an algorithm for automatically assessing spoken translations produced by language learners. The evaluation results demonstrated that our method involving deep linguistic analysis of the translation pair can achieve high consistency with human decisions, and our strategy of incorporating contextual constraints is effective at improving speech recognition on non-native speech. While our solution is domain specific, we emphasize domain portability in the linguistic analysis modules, so that similar capabilities in other domains can be quickly developed even in the absence of training data. Our interlingua framework also makes the methodology agnostic to the direction of source and target languages. A similar application for native Mandarin speakers learning English could be instantiated by using the same components for linguistic analysis.

A major challenge in our problem is in determining equivalence between the meanings of a translation pair. While our approach of using a rule-based generation system gives the developer great flexibility in deriving an appropriate KV representation, the comparison algorithm is somewhat primitive: it relies entirely on the generation rules to produce the right KV representation. In future work, we plan to apply machine learning techniques to this problem. With the data we have collected and labelled (and the effort is ongoing), it becomes feasible to examine the use of data-driven methods. As alluded to in our evaluation methodology, we can cast the problem into a classification framework. Lexical, *n*-gram, and alignment based features can be extracted from the translation pairs, which can be further enhanced by features obtained from deep linguistic analysis. This will relieve the burden on the semantic analysis component, and improve the overall portability of our approach.

We also plan to expand our application to many other domains appropriate for language learning, and test the effectiveness of the translation game as a means for language learning.

6 Acknowledgements

This research is supported in part by ITRI and the Cambridge MIT Initiative. The authors would like to acknowledge Yushi Xu for annotating the data. We are also grateful to Michael Collins and the anonymous reviews for their helpful comments and suggestions.

References

L. Baptist, S. Seneff. 2000. Genesis-II: A versatile system for language generation in conversational system applications. In *Proc. ICSLP*, Beijing, China.

D. Ehsani, E. Knodt. 1998. Speech technology in computer-aided language learnings: Strengths and limitations of a new call paradigm. *Language Learning & Technology*, 2(1):54–73.

M. Eskenazi. 1999. Using automatic speech processing for foreign language pronunciation tutoring: Some issues and a prototype. *Language Learning & Technology*, 2(2):62–76.

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

D. Hardison. 2004. Generalization of computer-assisted prosody training: quantitative and qualitative findings. *Language Learning & Technology*, 8(1):34–52.

E. Hovy, M. King, A. Popescu-Belis. 2002. Principles of context-based machine translation evaluation. *Machine Translation*, 7(1):43–75.

W. L. Johnson, S. Marsella, H. Vihjalmsson. 2004. The DARWARS tactical language training system. In *Proc. I/ITSEC*.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

S. Seneff, C. Wang, J. Zhang. 2004. Spoken conversational interaction for language learning. In *Proc. INSTIL/CALL*.

S. Seneff, C. Wang, J. Lee. 2006. Combining linguistic and statistical methods for bi-directional English Chinese translation in the flight domain. In *Proc. of AMTA*.

S. Seneff. 1992. TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1).

R. A. Solsona, E. Fosler-Lussier, H.-K. J. Kuo, A. Potamianos, I. Zitouni. 2002. Adaptive language models for spoken dialogue systems. In *ICASSP*.

J. S. Uebersax. 1998. Diversity of decision-making models and the measurement of interrater agreement. *Psychological Bulletin*, 101:140–146.

C. Wang, S. Seneff. 2006. High-quality speech translation in the flight domain. In *Proc. of InterSpeech*.

C. Wang, D. S. Cyphers, X. Mou, J. Polifroni, S. Seneff, J. Yi, V. Zue. 2000a. MUXING: A telephone-access Mandarin conversational system. In *Proc. ICSLP*, 715–718, Beijing, China.

H. C. Wang, F. Seide, C. Y. Tseng, L. S. Lee. 2000b. MAT2000 – Design, collection, and validation on a Mandarin 2000-speaker telephone speech database. In *Proc. ICSLP*, Beijing, China.

S. M. Witt. 1999. *Use of Speech Recognition in Computer-assisted Language Learning*. Ph.D. thesis, Department of Engineering, University of Cambridge, Cambridge, UK.

Automatic and human scoring of word definition responses

Kevyn Collins-Thompson and Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, U.S.A. 15213-8213
{kct|callan}@cs.cmu.edu

Abstract

Assessing learning progress is a critical step in language learning applications and experiments. In word learning, for example, one important type of assessment is a definition production test, in which subjects are asked to produce a short definition of the word being learned. In current practice, each free response is manually scored according to how well its meaning matches the target definition. Manual scoring is not only time-consuming, but also limited in its flexibility and ability to detect partial learning effects.

This study describes an effective automatic method for scoring free responses to definition production tests. The algorithm compares the text of the free response to the text of a reference definition using a statistical model of text semantic similarity that uses Markov chains on a graph of individual word relations. The model can take advantage of both corpus- and knowledge-based resources. Evaluated on a new corpus of human-judged free responses, our method achieved significant improvements over random and cosine baselines in both rank correlation and label error.

1 Introduction

Human language technologies are playing an increasingly important role in the science and practice of language learning. For example, intelligent Computer Assisted Language Learning (CALL) systems are being developed that can automatically tailor lessons and questions to the needs of individual students (Heilman et al., 2006). One critical task that language tutors, word learning experiments, and related applications have in common is assessing the learning progress of the student or experiment subject during the course of the session.

When the task is learning new vocabulary, a variety of tests have been developed to measure word learning progress. Some tests, such as multiple-choice selection of a correct synonym or cloze completion, are relatively passive. In production tests, on the other hand, students are asked to write or say a short phrase or sentence that uses the word being learned, called the *target word*, in a specified way.

In one important type of production test, called a *definition production* test, the subject is asked to describe the meaning of the target word, as they understand it at that point in the session. The use of such tests has typically required a teacher or researcher to manually score each response by judging its similarity in meaning to the reference definition of the target word. The resulting scores can then be used to analyze how a person's learning of the word responded to different stimuli, such as seeing the word used in context. A sample target word and its reference definition, along with examples of human-judged responses, are given in Sections 3.3 and 4.1.

However, manual scoring of the definition responses has several drawbacks. First, it is time-consuming and must be done by trained experts. Moreover, if the researcher wanted to test a new hy-

Proceedings of NAACL HLT 2007, pages 476–483,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

pothesis by examining the responses with respect to a different but related definition, the entire set of responses would have to be manually re-scored against the new target. Second, manual scoring can often be limited in its ability to detect when partial learning has taken place. This is due to the basic tradeoff between the sophistication of the graded scoring scale, and the ease and consistency with which human judges can use the scale. For example, it may be that the subject did not learn the complete meaning of a particular target word, but *did* learn that this target word had negative connotations. The usual binary or ternary score would provide no or little indication of such effects. Finally, because manual scoring almost always must be done off-line after the end of the session, it presents an obstacle to our goal of creating learning systems that can adapt quickly, within a single learning session.

This study describes an effective automated method for assessing word learning by scoring free responses to definition production tests. The method is flexible: it can be used to analyze a response with respect to whatever reference target(s) the teacher or researcher chooses. Such a test represents a powerful new tool for language learning research. It is also a compelling application of human language technologies research on semantic similarity, and we review related work for that area in Section 2. Our probabilistic model for computing text semantic similarity, described in Section 3, can use both corpus-based and knowledge-based resources. In Section 4 we describe a new dataset of human definition judgments and use it to measure the effectiveness of the model against other measures of text similarity. Finally, in Section 5 we discuss further directions and applications of our work.

2 Related Work

The problem of judging a subject response against a target definition is a type of text similarity problem. Moreover, it is a text *semantic* similarity task, since we require more than measuring direct word overlap between the two text fragments. For example, if the definition of the target word *ameliorate* is *to improve something* and the subject response is *make it better*, the response clearly indicates that the subject knows the meaning of the word, and thus should receive a high score, even though the response and the target definition have no words in common.

Because most responses are short (1 – 10 words) our task falls somewhere between word-word similarity and passage similarity. There is a broad field of existing work in estimating the semantic similarity of individual words. This field may be roughly divided into two groups. First, there are corpus-based measures, which use statistics or models derived from a large training collection. These require little or no human effort to construct, but are limited in the richness of the features they can reliably represent. Second, there are knowledge-based measures, which rely on specialized resources such as dictionaries, thesauri, experimental data, WordNet, and so on. Knowledge-based measures tend to be complementary to a corpus-based approach and emphasize precision in favor of recall. This is discussed further, along with a good general summary of text semantic similarity work, by (Mihalcea et al., 2006).

Because of the fundamental nature of the semantic similarity problem, there are close connections with other areas of human language technologies such as information retrieval (Salton and Lesk, 1971), text alignment in machine translation (Jayaraman and Lavie, 2005), text summarization (Mani and Maybury, 1999), and textual coherence (Foltz et al., 1998). Educational applications include automated scoring of essays, surveyed in (Valenti et al., 2003), and assessment of short-answer free-response items (Burstein et al., 1999).

As we describe in Section 3, we use a graph to model relations between words to perform a kind of *semantic smoothing* on the language models of the subject response and target definition before comparing them. Several types of relation, such as synonymy and co-occurrence, may be combined to model the interactions between terms. (Cao et al., 2005) also formulated a term dependency model combining multiple term relations in a language modeling framework, applied to information retrieval. Our graph-based approach may be viewed as a probabilistic variation on the *spreading activation* concept, originally proposed for word-word semantic similarity by (Quillian, 1967).

Finally, (Mihalcea et al., 2006) describe a text semantic similarity measure that combines word-word similarities between the passages being compared.

Due to limitations in the knowledge-based similarity measures used, semantic similarity is only estimated between words with the same part-of-speech. Our graph-based approach can relate words of different types and does not have this limitation. (Mihalcea et al., 2006) also evaluate their method in terms of paraphrase recognition using binary judgments. We view our task as somewhat different than paraphrase recognition. First, our task is not symmetric: we do not expect the target definition to be a paraphrase of the subject's free response. Second, because we seek sensitive measures of learning, we want to distinguish a range of semantic differences beyond a binary yes/no decision.

3 Statistical Text Similarity Model

We start by describing relations between pairs of terms using a general probability distribution. These pairs can then combine into a graph, which we can apply to define a semantic distance between terms.

3.1 Relations between individual words

One way to model word-to-word relationships is using a mixture of links, where each link defines a particular type of relationship. In a graph, this may be represented by a pair of nodes being joined by multiple weighted edges, with each edge corresponding to a different link type. Our link-based model is partially based on one defined by (Toutanova et al., 2004) for prepositional attachment. We allow directed edges because some relationships such as hypernyms may be asymmetric. The following are examples of different types of links.

1. **Stemming**: Two words are based on common morphology. Example: *stem* and *stemming*. We used Porter stemming (Porter, 1980).

2. **Synonyms and near-synonyms**: Two words share practically all aspects of meaning. Example: *quaff* and *drink*. Our synonyms came from WordNet (Miller, 1995).

3. **Co-occurrence**. Both words tend to appear together in the same contexts. Example: *politics* and *election*.

4. **Hyper- and hyponyms**: Relations such as "X is a kind of Y", as obtained from Wordnet or other thesaurus-like resources. Example: *airplane* and *transportation*.

5. **Free association**: A relation defined by the fact that a person is likely to give one word as a free-association response to the other. Example: *disaster* and *fear*. Our data was obtained from the Univ. of South Florida association database (Nelson et al., 1998).

We denote link functions using $\lambda_1, \ldots, \lambda_m$ to summarize different types of interactions between words. Each $\lambda_m(w_i, w_j)$ represents a specific type of lexical or semantic relation or constraint between w_i and w_j. For each link λ_m, we also define a weight γ_m that gives the strength of the relationship between w_i and w_j for that link.

Our goal is to predict the likelihood of a target definition $\mathcal{D}$ given a test response $\mathcal{R}$ consisting of terms $\{w_0 \ldots w_k\}$ drawn from a common vocabulary $\mathcal{V}$. We are thus interested in the conditional distribution $p(\mathcal{D} \mid \mathcal{R})$. We start by defining a simple model that can combine the link functions in a general purpose way to produce the conditional distribution $p(w_i|w_j)$ given arbitrary terms w_i and w_j. We use a log-linear model of the general form

$$p(w_i|w_j) = \frac{1}{Z} \exp \sum_{m=0}^{L} \gamma_m(i)\lambda_m(w_i, w_j) \quad (1)$$

In the next sections we show how to combine the estimate of individual pairs $p(w_i|w_j)$ into a larger graph of term relations, which will enable us to calculate the desired $p(\mathcal{D} \mid \mathcal{R})$.

3.2 Combining term relations using graphs

Graphs provide one rich model for representing multiple word relationships. They can be directed or undirected, and typically use nodes of words, with word labels at the vertices, and edges denoting word relationships. In this model, the dependency between two words represents a single inference step in which the label of the destination word is inferred from the source word. Multiple inference steps may then be chained together to perform longer-range inference about word relations. In this way, we can infer the similarity of two terms without requiring direct evidence for the relations between that specific pair. Using the link functions defined in Section 3.1, we imagine a generative process where an author A creates a short text of N words as follows.

Step 0: Choose an initial word w_0 with probability $P(w_0|A)$. (If we have already generated N words, stop.)

Step i**:** Given we have chosen w_{i-1}, then with probability $1-\alpha$ output the word w_{i-1} and reset the process to step 0. Otherwise, with probability α, sample a new word w_i according to the distribution:

$$P(w_i|w_{i-1}) = \frac{1}{Z} \exp \sum_{m=0}^{L} \gamma_m(i)\lambda_m(w_i, w_{i-1}) \tag{2}$$

where Z is the normalization quantity.

This conditional probability may be interpreted as a mixture model in which a particular link type $\lambda_m(.)$ is chosen with probability $\gamma_m(i)$ at timestep i. Note that the mixture is allowed to change at each timestep. For simplicity, we limit the number of such changes by grouping the timesteps of the walk into three stages: early, middle and final. The function $\Gamma(i)$ defines how timestep i maps to stage s, where $s \in \{0, 1, 2\}$, and we now refer to $\gamma_m(s)$ instead of $\gamma_m(i)$.

Suppose we have a definition $\mathcal{D}$ consisting of terms $\{d_i\}$. For each link type $\lambda_m(.)$ we define a transition matrix $C(\mathcal{D}, m)$ based on the definition D. The reason $\mathcal{D}$ influences the transition matrix is that some link types, such as proximity and co-occurrence, are context-specific. Each stage s has an overall transition matrix $C(\mathcal{D}, s)$ as the mixture of the individual $C(\mathcal{D}, m)$, as follows.

$$C(\mathcal{D}, s) = \sum_{m=1}^{M} \gamma_m(s) C(\mathcal{D}, m) \tag{3}$$

Combining the stages over k steps into a single transition matrix, which we denote C^k, we have

$$C^k = \prod_{i=0}^{k} C(\mathcal{D}, \Gamma(i)) \tag{4}$$

We denote the (i, j) entry of a matrix A^k by $A^k_{i,j}$. Then for a particular term d_i, the probability that a chain reaches d_i after k steps, starting at word w is

$$P_k(d_i|w) = (1-\alpha)\alpha^k C^k_{w,d_i} \tag{5}$$

where we identify w and d_i with their corresponding indices into the vocabulary $\mathcal{V}$. The overall probability $p(d_i|w)$ of generating a definition term d_i given a word w is therefore

$$P(d_i|w) = \sum_{k=0}^{\infty} P_k(d_i|w) = (1-\alpha)(\sum_{k=0}^{\infty} \alpha^k C^k)_{w,d_i} \tag{6}$$

The walk continuation probability α can be viewed as a penalty for long chains of inference. In practice, to perform the random walk steps we replace the infinite sum of Eq. 6 with a small number of steps (up to 5) on a sparse representation of the adjacency graph. We obtained effective link weights $\gamma_m(i)$ empirically using held-out data. For simplicity we assume that the same α is used across all link types, but further improvement may be possible by extending the model to use link-specific decays α_m. Fine-tuning these parameter estimation methods is a subject of future work.

3.3 Using the model for definition scoring

In our study the reference definition for the target word consisted of the target word, a rare synonym, a more frequent synonym, and a short glossary-like definition phrase. For example, the reference definition for *abscond* was

abscond; absquatulate; escape; to leave quickly and secretly and hide oneself, often to avoid arrest or prosecution.

In general, we define the score of a response $\mathcal{R}$ with respect to a definition $\mathcal{D}$ as the probability that the definition is generated by the response, or $p(\mathcal{D}|\mathcal{R})$. Equivalently, we can score by $\log p(\mathcal{D}|\mathcal{R})$ since the $\log$ function is monotonic. So making the simplifying assumption that the terms $d_i \in D$ are exchangable (the bag-of-words assumption), and taking logarithms, we have:

$$\begin{aligned} \log p(\mathcal{D}|\mathcal{R}) &= \log \prod_{d_i \in \mathcal{D}} p(d_i|\mathcal{R}) \\ &= \sum_{d_i \in \mathcal{D}} \log[(1-\alpha)(\sum_{k=0}^{m} \alpha^k C^k)_{\mathcal{R},d_i}] \end{aligned} \tag{7}$$

Suppose that the response to be scored is *run from the cops*. In practical terms, Eq. 7 means that for our example, we "light up" the nodes in the graph corresponding to *run, from, the* and *cops* by assigning some initial probability, and the graph is then "run" using the transition matrix C according to Eq. 7. In this study, the initial node probabilities are set to values proportional to the *idf* values of the corresponding term, so that $P(d_i) = \frac{idf(d_i)}{\sum idf(d_i)}$. After m steps, the probabilities at the nodes for each term in the reference definition $\mathcal{R}$ are read off, and their logarithms summed. Similar to an AND calculation, we calculate a product of sums over the graph, so that responses reflecting multiple aspects of the target definition are rewarded more highly than a very strong prediction for only a single definition term.

4 Evaluation

We first describe our corpus of gold standard human judgments. We then explain the different text similarity methods and baselines we computed on the corpus responses. Finally, we give an analysis and discussion of the results.

4.1 Corpus

We obtained a set of 734 responses to definition production tests from a word learning experiment at the University of Pittsburgh (Bolger et al., 2006). In total, 72 target words, selected by the same group, were used in the experiment. In this experiment, subjects were asked to learn the meaning of target words after seeing them used in a series of context sentences. We set aside 70 responses for training, leaving 664 responses in the final test dataset.

Each response instance was coded using the scale shown in Table 1, and a sample set of subject responses and scores is shown in Table 2. The target word was treated as having several key aspects of meaning. The coders were instructed to judge a response according to how well it covered the various aspects of the target definition. If the response covered all aspects of the target definition, but also included extra irrelevant information, this was treated as a partial match at the discretion of the coders.

We obtained three codings of the final dataset. The first two codings were obtained using an independent group, the QDAP Center at the University of Pittsburgh. Initially, five human coders, with varying degrees of general coding experience, were trained by the authors using one set of 10 example instances and two training sessions of 30 instances each. Between the two training sessions, one of the authors met with the coders to discuss the ratings and refine the rating guidelines. After training, the authors selected the two coders who had the best inter-coder agreement on the 60 training instances. These two coders then labeled the final test set of 664 instances. Our third coding was obtained from an initial coding created by an expert in the University of Pittsburgh Psychology department and then adjusted by one of the authors to resolve a small number of internal inconsistencies, such as when the same response to the same target had been given a different score.

Inter-coder agreement was measured using linear weighted kappa, a standard technique for ordinal scales. Weighted kappa scores for all three coder pairs are shown in Table 3. Overall, agreement ranged from moderate (0.64) to good (0.72).

Score	Meaning
0	Completely wrong
1	Some partial aspect is correct
2	One major aspect, or more than one minor aspect, is correct
3	Covers all aspects correctly

Table 1: Scale for human definition judgements.

Response	Human Score
depart secretly	3
quietly make away, escape	3
to flee, run away	2
flee	2
to get away with	1
to steal or take	0

Table 2: Examples of human scores of responses for the target word *abscond*.

4.2 Baseline Methods

We computed three baselines as reference points for lower and upper performance bounds.

Random. The response items were assigned labels randomly.

Coder pair	Weighted Kappa
1, 2	0.68
2, 3	0.64
1, 3	0.72

Table 3: Weighted kappa inter-rater reliability for three human coders on our definition response dataset (664 items).

Method	Spearman Rank Correlation
Random	0.3661
Cosine	0.4731
LSA	0.4868
Markov	0.6111
LSA + Markov	0.6365
Human	0.8744

Table 4: Ability of methods to match human ranking of responses, as measured by Spearman rank correlation (corrected for ties).

Human choice of label. We include a method that, given an item and a human label from one of the coders, simply returns a label of the same item from a different coder, with results repeated and averaged over all coders. This gives an indication of an upper bound based on human performance.

Cosine similarity using tf.idf weighting. Cosine similarity is a widely-used text similarity method for tasks where the passages being compared often have significant direct word overlap. We represent response items and reference definitions as vectors of terms using *tf.idf* weighting, a standard technique from information retrieval (Salton and Buckley, 1997) that combines term frequency (*tf*) with term specificity (*idf*). A good summary of arguments for using *idf* can be found in (Robertson, 2004). To compute *idf*, we used frequencies from a standard 100-million-word corpus of written and spoken English [1]. We included a minimal semantic similarity component by applying Porter stemming (Porter, 1980) on terms.

[1]The British National Corpus (Burnage and Dunlop, 1992), using American spelling conversion.

4.3 Methods

In addition to the baseline methods, we also ran the following three algorithms over the responses.

Markov chains ("Markov"). This is the method described in Section 3. A maximum of 5 random walk steps were used, with a walk continuation probability of 0.8. Each walk step used a mixture of synonym, stem, co-occurrence, and free-association links. The link weights were trained on a small set of held-out data.

Latent Semantic Analysis (LSA). LSA (Landauer et al., 1998) is a corpus-based unsupervised technique that uses dimensionality reduction to cluster terms according to multi-order co-occurrence relations. In these experiments, we obtained LSA-based similarity scores between responses and target definitions using the software running on the University of Colorado LSA Web site (LSA site, 2006). We used the pairwise text passage comparison facility, using the maximum 300 latent factors and a general English corpus (Grade 1 – first-year college).

Although LSA and the Markov chain approach are based on different principles, we chose to apply LSA to this new response-scoring task and corpus because LSA has been widely used as a text semantic similarity measure for other tasks and shown good performance (Foltz et al., 1998).

LSA+Markov. To test the effectiveness of combining two different – and possibly complementary – approaches to response scoring, we created a normalized, weighted linear combination of the LSA and Markov scores, with the model combination weight being derived from cross-validation on a held-out dataset.

4.4 Results

We measured the effectiveness of each scoring method from two perspectives: ranking quality, and label accuracy.

First, we measured how well each scoring method was able to rank response items by similarity to the target definition. To do this, we calculated the Spearman Rank Correlation (corrected for ties) between the ranking based on the scoring method and the ranking based on the human-assigned scores, averaged over all sets of target word responses.

Table 4 summarizes the ranking results. For

Method	Label error (RMS)	
	Top 1	Top 3
Random	1.4954	1.6643
Cosine	0.8194	1.0540
LSA	0.8009	0.9965
Markov	0.7222	0.7968
LSA + Markov	1.1111	1.0650
Human	0.1944	0.4167

Table 5: Root mean squared error (RMSE) of label(s) for top-ranked item, and top-three items for all 77 words in the dataset.

overall quality of ranking, the Markov method had significantly better performance than the other automated methods ($p < 2.38e^{-5}$). LSA gave a small, but not significant, improvement in overall rank quality over the cosine baseline. [2] The simple combination of LSA and Markov resulted in a slightly higher but statistically insignificant difference ($p < 0.253$).

Second, we examined the ability of each method to find the most accurate responses – that is, the responses with the highest human label on average – for a given target word. To do this, we calculated the Root Mean Squared Error (RMSE) of the label assigned to the top item, and the top three items. The results are shown in Table 5. For top-item detection, our Markov model had the lowest RMS error (0.7222) of the automated methods, but the differences from Cosine and LSA were not statistically significant, while differences for all three from Random and Human baselines were significant. For the top three items, the difference between Markov (0.7968) and LSA (0.9965) was significant at the $p < 0.03$ level.

Comparing the overall rank accuracy with top-item accuracy, the combined LSA + Markov method was significantly worse at finding the three best-quality responses (RMSE of 1.0650) than Markov (0.7968) or LSA (0.9965) alone. The reasons for this require further study.

[2]All statistical significance results reported here used the Wilcoxon Signed-Ranks test.

5 Discussion

Even though definition scoring may seem more straightforward than other automated learning assessment problems, human performance was still significantly above the best automated methods in our study, for both ranking and label accuracy. There are certain additions to our model which seem likely to result in further improvement.

One of the most important is the ability to identify phrases or colloquial expressions. Given the short length of a response, these seem critical to handle properly. For example, *to get away with something* is commonly understood to mean *secretly guilty*, not a physical action. Yet the near-identical phrase *to get away from something* means something very different when phrases and idioms are considered.

Despite the gap between human and automated performance, the current level of accuracy of the Markov chain approach has already led to some promising early results in word learning research. For example, in a separate study of incremental word learning (Frishkoff et al., 2006), we used our measure to track increments in word knowledge across multiple trials. Each trial consisted of a single passage that was either *supportive* – containing clues to the meaning of unfamiliar words – or not supportive. In this separate study, broad learning effects identified by our measure were consistent with effects found using manually-scored pre- and post-tests. Our automated method also revealed a previously unknown interaction between trial spacing, the proportion of supportive contexts per word, and reader skill.

In future applications, we envision using our automated measure to allow a form of feedback for intelligent language tutors, so that the system can automatically adapt its behavior based on the student's test responses. With some adjustments, the same scoring model described in this study may also be applied to the problem of finding supportive contexts for students.

6 Conclusions

We presented results for both automated and human performance of an important task for language learning applications: scoring definition responses. We described a probabilistic model of text semantic similarity that uses Markov chains on a graph of term relations to perform a kind of semantic smoothing. This model incorporated both corpus-based and knowledge-based resources to compute text semantic similarity. We measured the effectiveness of both our method and LSA compared to cosine and random baselines, using a new corpus of human judgments on definition responses from a language learning experiment. Our method outperformed the *tf.idf* cosine similarity baseline in ranking quality and in ability to find high-scoring definitions. Because LSA and our Markov chain method are based on different approaches and resources, it is difficult to draw definitive conclusions about performance differences between the two methods.

Looking beyond definition scoring, we believe automated methods for assessing word learning have great potential as a new scientific tool for language learning researchers, and as a key component of intelligent tutoring systems that can adapt to students.

Acknowledgements

We thank D.J. Bolger and C. Perfetti for the use of their definition response data, Stuart Shulman for his guidance of the human coding effort, and the anonymous reviewers for their comments. This work supported by U.S. Dept. of Education grant R305G03123. Any opinions, findings, and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

D.J. Bolger, M. Balass, E. Landen and C.A. Perfetti. 2006. Contextual Variation and Definitions in Learning the Meanings of Words. (In press.)

G. Burnage and D. Dunlop. 1992. Encoding the British National Corpus. *English Language Corpora: Design, Analysis and Exploitation*. The 13th Intl. Conf. on Engl. Lang. Res. in Computerized Corpora. Nijmegen. J. Aarts, P. de Haan, N. Oostdijk, Eds.

J. Burstein, S. Wolff, and L. Chi. 1999. Using Lexical Semantic Techniques to Classify Free-Responses. *Breadth and Depth of Semantic Lexicons*. Kluwer Acad. Press, p. 1–18.

G. Cao, J-Y. Nie, and J. Bai. Integrating Word Relationships into Language Models. *SIGIR 2005*, 298–305.

P.W. Foltz, W. Kintsch, and T. Landauer. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes,* 25(2):285–307.

G. Frishkoff, K. Collins-Thompson, J. Callan, and C. Perfetti. 2007. The Nature of Incremental Word Learning: Context Quality, Spacing Effects, and Skill Differences in Meaning Acquisition Across Multiple Contexts. (In preparation.)

M. Heilman, K. Collins-Thompson, J. Callan and M. Eskanazi. 2006. Classroom Success of an Intelligent Tutoring System for Lexical Practice and Reading Comprehension. *ICSLP 2006.*

S. Jayaraman and A. Lavie. Multi-Engine Machine Translation Guided by Explicit Word Matching. *EAMT 2005.*

T.K. Landauer, P.W. Foltz, and D. Laham. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes,* 25:259–284.

LSA Web Site. http://lsa.colorado.edu

I. Mani and M.T. Maybury (Eds.) 1999. Advances in Automatic Text Summarization. MIT Press.

R. Mihalcea, C. Corley, and C. Strapparava. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. *AAAI 2006*

G. Miller. 1998. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11) 39–41.

D.L. Nelson, C.L. McEvoy, and T.A. Schreiber. 1998. The University of South Florida word association, rhyme, and word fragment norms. http://www.usf.edu/FreeAssociation/

M. Porter. 1980. An Algorithm for Suffix-stripping. *Program,* 14(3) 130–137. http://www.tartarus.org/martin/PorterStemmer

M. Quillian. 1967. Word Concepts: A Theory and Simulation of some Basic Semantic Capabilities. *Behav. Sci.*, 12: 410–430.

S. Robertson. 2004. Understanding Inverse Document Frequency: on Theoretical Arguments for IDF. *J. of Documentation*, 60:503–520.

G. Salton and C. Buckley. 1997. Term Weighting Approaches in Automatic Text Retrieval. *Reading in Information Retrieval*. Morgan Kaufmann.

G. Salton and M. Lesk. 1971. *Computer Evaluation of Indexing and Text Processing*. Prentice-Hall. 143 – 180.

K. Toutanova, C.D. Manning, and A.Y. Ng. 2004. Learning Random Walk Models for Inducing Word Dependency Distributions. *ICML 2004.*

S. Valenti, F. Neri, and A. Cucchiarelli. 2003. An Overview of Current Research on Automated Essay Grading. *J. of Info. Tech. Ed.*, Vol. 2.

A Comparison of Pivot Methods for Phrase-based Statistical Machine Translation

Masao Utiyama and **Hitoshi Isahara**
National Institute of Information and Communications Technology
3-5 Hikari-dai, Soraku-gun, Kyoto 619-0289 Japan
{mutiyama,isahara}@nict.go.jp

Abstract

We compare two pivot strategies for phrase-based statistical machine translation (SMT), namely *phrase translation* and *sentence translation*. The phrase translation strategy means that we directly construct a phrase translation table (phrase-table) of the source and target language pair from two phrase-tables; one constructed from the source language and English and one constructed from English and the target language. We then use that phrase-table in a phrase-based SMT system. The sentence translation strategy means that we first translate a source language sentence into n English sentences and then translate these n sentences into target language sentences separately. Then, we select the highest scoring sentence from these target sentences. We conducted controlled experiments using the Europarl corpus to evaluate the performance of these pivot strategies as compared to directly trained SMT systems. The phrase translation strategy significantly outperformed the sentence translation strategy. Its relative performance was 0.92 to 0.97 compared to directly trained SMT systems.

1 Introduction

The rapid and steady progress in corpus-based machine translation (Nagao, 1981; Brown et al., 1993) has been supported by large parallel corpora such as the Arabic-English and Chinese-English parallel corpora distributed by the Linguistic Data Consortium and the Europarl corpus (Koehn, 2005), which consists of 11 European languages. However, large parallel corpora do not exist for many language pairs. For example, there are no publicly available Arabic-Chinese large-scale parallel corpora even though there are Arabic-English and Chinese-English parallel corpora.

Much work has been done to overcome the lack of parallel corpora. For example, Resnik and Smith (2003) propose mining the web to collect parallel corpora for low-density language pairs. Utiyama and Isahara (2003) extract Japanese-English parallel sentences from a noisy-parallel corpus. Munteanu and Marcu (2005) extract parallel sentences from large Chinese, Arabic, and English non-parallel newspaper corpora.

Researchers can also make the best use of existing (small) parallel corpora. For example, Nießen and Ney (2004) use morpho-syntactic information to take into account the interdependencies of inflected forms of the same lemma in order to reduce the amount of bilingual data necessary to sufficiently cover the vocabulary in translation. Callison-Burch et al. (2006a) use paraphrases to deal with unknown source language phrases to improve coverage and translation quality.

In this paper, we focus on situations where no parallel corpus is available (except a few hundred parallel sentences for tuning parameters). To tackle these extremely scarce training data situations, we propose using a pivot language (English) to bridge the source and target languages in translation. We first translate source language sentences or phrases into English and then translate those English sentences or phrases into the target language, as described in Section 3. We thus assume that there is a parallel corpus consisting of the source language and English as well as one consisting of English and the target language. Selecting English as a pivot language is a reasonable pragmatic choice because English is included in parallel corpora more often than other languages are, though any language can be used as a pivot language.

In Section 2, we describe a phrase-based statistical machine translation (SMT) system that was used to develop the pivot methods described in Section 3. This is the shared task baseline system for the 2006 NAACL/HLT workshop on statistical machine translation (Koehn and Monz, 2006) and consists of the Pharaoh decoder (Koehn, 2004), SRILM (Stolcke, 2002), GIZA++ (Och and Ney, 2003), mkcls (Och, 1999), Carmel,[1] and a phrase model training code.

2 Phrase-based SMT

We use a phrase-based SMT system, Pharaoh, (Koehn et al., 2003; Koehn, 2004), which is based on a log-linear formulation (Och and Ney, 2002). It is a state-of-the-art SMT system with freely available software, as described in the introduction. The system segments the source sentence into so-called phrases (a number of sequences of consecutive words). Each phrase is translated into a target language phrase. Phrases may be reordered.

Let $\mathbf{f}$ be a source sentence (e.g, French) and $\mathbf{e}$ be a target sentence (e.g., English), the SMT system outputs an $\hat{\mathbf{e}}$ that satisfies

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) \quad (1)$$

$$= \arg\max_{\mathbf{e}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f}) \quad (2)$$

where $h_m(\mathbf{e}, \mathbf{f})$ is a feature function and λ_m is a weight. The system uses a total of eight feature functions: a trigram language model probability of the target language, two phrase translation probabilities (both directions), two lexical translation probabilities (both directions), a word penalty, a phrase penalty, and a linear reordering penalty. For details on these feature functions, please refer to (Koehn et al., 2003; Koehn, 2004; Koehn et al., 2005). To set the weights, λ_m, we carried out minimum error rate training (Och, 2003) using BLEU (Papineni et al., 2002) as the objective function.

[1] `http://www.isi.edu/licensed-sw/carmel/`

3 Pivot methods

We use the phrase-based SMT system described in the previous section to develop pivot methods. We use English $\mathbf{e}$ as the pivot language. We use French $\mathbf{f}$ and German $\mathbf{g}$ as examples of the source and target languages in this section.

We describe two types of pivot strategies, namely *phrase translation* and *sentence translation*.

The phrase translation strategy means that we directly construct a French-German phrase translation table (phrase-table for short) from a French-English phrase-table and an English-German phrase-table. We assume that these French-English and English-German tables are built using the phrase model training code in the baseline system described in the introduction. That is, phrases are heuristically extracted from word-level alignments produced by doing GIZA++ training on the corresponding parallel corpora (Koehn et al., 2003).

The sentence translation strategy means that we first translate a French sentence into n English sentences and translate these n sentences into German separately. Then, we select the highest scoring sentence from the German sentences.

3.1 Phrase translation strategy

The phrase translation strategy is based on the fact that the phrase-based SMT system needs a phrase-table and a language model for translation. Usually, we have the language model of a target language. Consequently, we only need to construct a phrase-table to train the phrase-based SMT system.

We assume that we have a French-English phrase-table T_{FE} and an English-German phrase-table T_{EG}. From these tables, we construct a French-German phrase-table T_{FG}, which requires estimating four feature functions; phrase translation probabilities for both directions, $\phi(\bar{f}|\bar{g})$ and $\phi(\bar{g}|\bar{f})$ and lexical translation probabilities for both directions,

$p_w(\bar{f}|\bar{g})$ and $p_w(\bar{g}|\bar{f})$, where $\bar{f}$ and $\bar{g}$ are French and German phrases that are parts of phrase translation pairs in T_{FE} and T_{EG}, respectively.[2]

We estimate these probabilities using the probabilities available in T_{FE} and T_{EG} as follows.[3]

$$\phi(\bar{f}|\bar{g}) = \sum_{\bar{e} \in T_{FE} \cap T_{EG}} \phi(\bar{f}|\bar{e})\phi(\bar{e}|\bar{g}) \quad (3)$$

$$\phi(\bar{g}|\bar{f}) = \sum_{\bar{e} \in T_{FE} \cap T_{EG}} \phi(\bar{g}|\bar{e})\phi(\bar{e}|\bar{f}) \quad (4)$$

$$p_w(\bar{f}|\bar{g}) = \sum_{\bar{e} \in T_{FE} \cap T_{EG}} p_w(\bar{f}|\bar{e})p_w(\bar{e}|\bar{g}) \quad (5)$$

$$p_w(\bar{g}|\bar{f}) = \sum_{\bar{e} \in T_{FE} \cap T_{EG}} p_w(\bar{g}|\bar{e})p_w(\bar{e}|\bar{f}) \quad (6)$$

where $\bar{e} \in T_{FE} \cap T_{EG}$ means that the English phrase $\bar{e}$ is included in both T_{FE} and T_{EG} as part of phrase translation pairs. $\phi(\bar{f}|\bar{e})$ and $\phi(\bar{e}|\bar{f})$ are phrase translation probabilities for T_{FE} and $\phi(\bar{e}|\bar{g})$ and $\phi(\bar{g}|\bar{e})$ are those for T_{EG}. $p_w(\bar{f}|\bar{e})$ and $p_w(\bar{e}|\bar{f})$ are lexical translation probabilities for T_{FE} and $p_w(\bar{e}|\bar{g})$ and $p_w(\bar{g}|\bar{e})$ are those for T_{EG}.

The definitions of the phrase and lexical translation probabilities are as follows (Koehn et al., 2003).

$$\phi(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{f}, \bar{e})}{\sum_{\bar{f}'} \text{count}(\bar{f}', \bar{e})} \quad (7)$$

where $\text{count}(\bar{f}, \bar{e})$ gives the total number of times the phrase $\bar{f}$ is aligned with the phrase $\bar{e}$ in the parallel corpus. Eq. 7 means that $\phi(\bar{f}|\bar{e})$ is calculated using maximum likelihood estimation.

The definition of the lexical translation probability is

$$p_w(\bar{f}|\bar{e}) = \max_{\mathbf{a}} p_w(\bar{f}|\bar{e}, \mathbf{a}) \quad (8)$$

$$p_w(\bar{f}|\bar{e}, \mathbf{a}) = \prod_{i=1}^{n} E_w(f_i|\bar{e}, \mathbf{a}) \quad (9)$$

$$E_w(f_i|\bar{e}, \mathbf{a}) = \frac{1}{|\{j|(i,j) \in \mathbf{a}\}|} \sum_{\forall(i,j) \in \mathbf{a}} w(f_i|e_j) \quad (10)$$

$$w(f|e) = \frac{\text{count}(f, e)}{\sum_{f'} \text{count}(f', e)} \quad (11)$$

where $\text{count}(f, e)$ gives the total number of times the word f is aligned with the word e in the parallel corpus. Thus, $w(f|e)$ is the maximum likelihood estimation of the word translation probability of f given e. $Ew(f_i|\bar{e}, \mathbf{a})$ is calculated from a word alignment $\mathbf{a}$ between a phrase pair $\bar{f} = f_1 f_2 \ldots f_n$ and $\bar{e} = e_1 e_2 \ldots e_m$ where f_i is connected to several $(|\{j|(i,j) \in \mathbf{a}\}|)$ English words. Thus, $Ew(f_i|\bar{e}, \mathbf{a})$ is the average (or mixture) of $w(f_i|e_j)$. This means that $Ew(f_i|\bar{e}, \mathbf{a})$ is an estimation of the probability of f_i in $\mathbf{a}$. Consequently, $p_w(\bar{f}|\bar{e}, \mathbf{a})$ estimates the probability of $\bar{f}$ given $\bar{e}$ and $\mathbf{a}$ using the product of the probabilities $Ew(f_i|\bar{e}, \mathbf{a})$. This assumes that the probability of f_i is independent given $\bar{e}$ and $\mathbf{a}$. $p_w(\bar{f}|\bar{e})$ takes the highest $p_w(\bar{f}|\bar{e}, \mathbf{a})$ if there are multiple alignments $\mathbf{a}$. This discussion, which is partly based on Section 4.1.2 of (Och and Ney, 2004), means that the lexical translation probability $p_w(\bar{f}|\bar{e})$ is another probability estimated using the word translation probability $w(f|e)$.

The justification of Eqs. 3–6 is straightforward. From the discussion above, we know that the probabilities, $\phi(\bar{f}|\bar{e})$, $\phi(\bar{e}|\bar{f})$, $\phi(\bar{g}|\bar{e})$, $\phi(\bar{e}|\bar{g})$, $p_w(\bar{f}|\bar{e})$, $p_w(\bar{e}|\bar{f})$, $p_w(\bar{g}|\bar{e})$, and $p_w(\bar{e}|\bar{g})$ are probabilities in the ordinary sense. Thus, we can derive $\phi(\bar{f}|\bar{g})$, $\phi(\bar{g}|\bar{f})$, $p_w(\bar{f}|\bar{g})$, and $p_w(\bar{g}|\bar{f})$ by assuming that these probabilities are independent given an English phrase $\bar{e}$ (e.g., $\phi(\bar{f}|\bar{g}, \bar{e}) = \phi(\bar{f}|\bar{e})$).

We construct a T_{FG} that consists of all French-German phrases whose phrase and lexical translation probabilities as defined in Eqs. 3–6 are greater than 0. We use the term *PhraseTrans* to denote SMT systems that use the phrase translation strategy described above.

3.2 Sentence translation strategy

The sentence translation strategy uses two independently trained SMT systems. We first translate a French sentence $\mathbf{f}$ into n English sentences $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n$ using a French-English SMT system. Each $\mathbf{e}_i$ $(i = 1 \ldots n)$ has the eight scores calculated from the eight feature functions described in Section 2. We denote these scores $h^e_{i1}, h^e_{i2}, \ldots h^e_{i8}$. Second, we translate each $\mathbf{e}_i$ into n German sentences $\mathbf{g}_{i1}, \mathbf{g}_{i2}, \ldots, \mathbf{g}_{in}$ using an English-German

[2]Feature functions scores are calculated using these probabilities. For example, for a translation probability of a French sentence $\mathbf{f} = \bar{f}_1 \ldots \bar{f}_K$ and a German sentence $\mathbf{g} = \bar{g}_1 \ldots \bar{g}_K$, $h(\mathbf{g}, \mathbf{f}) = \log \prod_{i=1}^{K} \phi(\bar{f}_i|\bar{g}_i)$, where K is the number of phrases.

[3]Wang et al. (2006) use essentially the same definition to induce the translation probability of the source and target language word alignment that is bridged by an intermediate language. Callison-Burch et al. (2006a) use a similar definition for a paraphrase probability.

SMT system. Each $\mathbf{g}_{ij}$ $(j = 1 \dots n)$ has the eight scores, which are denoted as $h^g_{ij1}, h^g_{ij2}, \dots, h^g_{ij8}$. This situation is depicted as

$$\begin{array}{lll} \mathbf{f} & \rightarrow & \mathbf{e}_i \quad (h^e_{i1}, h^e_{i2}, \dots, h^e_{i8}) \\ & \rightarrow & \mathbf{g}_{ij} \quad (h^g_{ij1}, h^g_{ij2}, \dots, h^g_{ij8}) \end{array}$$

We define the score of $\mathbf{g}_{ij}$, $S(\mathbf{g}_{ij})$, as

$$S(\mathbf{g}_{ij}) = \sum_{m=1}^{8} (\lambda^e_m h^e_{im} + \lambda^g_m h^g_{ijm}) \qquad (12)$$

where λ^e_m and λ^g_m are weights set by performing minimum error rate training[4] as described in Section 2. We select the highest scoring German sentence

$$\hat{\mathbf{g}} = \arg\max_{\mathbf{g}_{ij}} S(\mathbf{g}_{ij}) \qquad (13)$$

as the translation of the French sentence $\mathbf{f}$.

A drawback of this strategy is that translation speed is about $O(n)$ times slower than those of the component SMT systems. This is because we have to run the English-German SMT system n times for a French sentence. Consequently, we cannot set n very high. When we used $n = 15$ in the experiments described in Section 4, it took more than two days to translate 3064 test sentences on a 3.06GHz LINUX machine.

Note that when $n = 1$, the above strategy produces the same translation with the simple sequential method that we first translate a French sentence into an English sentence and then translate that sentence into a German sentence.

We use the terms *SntTrans15* and *SntTrans1* to denote SMT systems that use the sentence translation strategy with $n = 15$ and $n = 1$, respectively.

4 Experiments

We conducted controlled experiments using the Europarl corpus. For each language pair described below, the Europarl corpus provides three types of parallel corpora; the source language–English, English–the target language, and the source language–the target language. This means that we can directly train an SMT system using the source and target language parallel corpus as well as pivot SMT systems using English as the pivot language. We use the term *Direct* to denote directly trained SMT systems. For each language pair, we compare four SMT systems; *Direct, PhraseTrans, SntTrans15,* and *SntTrans1*.[5]

4.1 Training, tuning and testing SMT systems

We used the training data for the shared task of the SMT workshop (Koehn and Monz, 2006) to train our SMT systems. It consists of three parallel corpora: French-English, Spanish-English, and German-English.

We used these three corpora to extract a set of sentences that were aligned to each other across all four languages. For that purpose, we used English as the pivot. For each distinct English sentence, we extracted the corresponding French, Spanish, and German sentences. When an English sentence occurred multiple times, we extracted the most frequent translation. For example, because "Resumption of the session" was translated into "Wiederaufnahme der Sitzungsperiode" 120 times and "Wiederaufnahme der Sitzung" once, we extracted "Wiederaufnahme der Sitzungsperiode" as its translation. Consequently, we extracted 585,830 sentences for each language. From these corpora, we constructed the training parallel corpora for all language pairs.

We followed the instruction of the shared task baseline system to train our SMT systems.[6] We used the trigram language models provided with the shared task. We did minimum error rate training on the first 500 sentences in the shared task development data to tune our SMT systems and used the

[4]We use a reranking strategy for the sentence translation strategy. We first obtain n^2 German sentences for each French sentence by applying two independently trained French-English and English-German SMT systems. Each of the translated German sentences has the sixteen scores as described above. The weights in Eq. 12 are tuned against reference German sentences by performing minimum error rate training. These weights are in general different from those of the original French-English and English-German SMT systems.

[5]As discussed in the introduction, we intend to use the pivot strategies in a situation where a very limited amount of parallel text is available. The use of the Europarl corpus is not an accurate simulation of the intended situation because it enables us to use a relatively large parallel corpus for direct training. However, it is necessary to evaluate the performance of the pivot strategies against that of *Direct* SMT systems under controlled experiments in order to determine how much the pivot strategies can be improved. This is a first step toward the use of pivot methods in situations where training data is extremely scarce.

[6]The parameters for the Pharaoh decoder were "-dl 4 -b 0.03 -s 100". The maximum phrase length was 7.

3064 test sentences for each language as our test set.

Our evaluation metric was %BLEU scores, as calculated by the script provided along with the shared task.[7] We lowercased the training, development and test sentences.

4.2 Results

Table 1 compares the BLEU scores of the four SMT systems; *Direct*, *PhraseTrans*, *SntTrans15*, and *SntTrans1* for each language pair. The columns SE and ET list the BLEU scores of the *Direct* SMT systems trained on the source language–English and English–the target language parallel corpora. The numbers in the parentheses are the relative scores of the pivot SMT systems, which were obtained by dividing their BLEU scores by that of the corresponding *Direct* system. For example, for the Spanish–French language pair, the BLEU score of the *Direct* SMT system was 35.78, that of the *PhraseTrans* SMT system was 32.90, and the relative performance was $0.92 = (32.90/35.78)$. For the *SntTrans15* SMT system, the BLEU score was 29.49 and the relative performance was $0.82 = (29.49/35.78)$.

The BLEU scores of the *Direct* SMT systems were higher than those of the *PhraseTrans* SMT systems for all six source-target language pairs. The *PhraseTrans* SMT systems performed better than the *SntTrans15* SMT systems for all pairs. The *SntTrans15* SMT systems were better than the *SntTrans1* SMT systems for four pairs. According to the sign test, under the null hypothesis that the BLEU scores of two systems are equivalent, finding one system obtaining better BLEU scores on all six language pairs is statistically significant at the 5 % level. Obtaining four better scores is not statistically significant. Thus, Table 1 indicates

$$\textit{Direct} > \textit{PhraseTrans} > \textit{SntTrans15} \sim \textit{SntTrans1}$$

where "$>$" and "$\sim$" means that the differences of the BLEU scores of the corresponding SMT systems are statistically significant and insignificant, respectively.

[7]Callison-Burch et al. (2006b) show that in general a higher BLEU score is not necessarily indicative of better translation quality. However, they also suggest that the use of BLEU is appropriate for comparing systems that use similar translation strategies, which is the case with our experiments.

As expected, the *Direct* SMT systems outperformed the other systems. We regard the BLEU scores of the *Direct* systems as the upperbound. The *SntTrans15* SMT systems did not significantly outperform the *SntTrans1* SMT systems. We think that this is because $n = 15$ was not large enough to cover good translation candidates.[8] Selecting the highest scoring translation from a small pool did not always lead to better performance. To improve the performance of the sentence translation strategy, we need to use a large n. However, this is not practical because of the slow translation speed, as discussed in Section 3.2.

The *PhraseTrans* SMT systems significantly outperformed the *SntTrans15* and *SntTrans1* systems. That is, the phrase translation strategy is better than the sentence translation strategy. Since the phrase-tables constructed using the phrase translation strategy can be integrated into the Pharaoh decoder as well as the directly extracted phrase-tables, the *PhraseTrans* SMT systems can fully exploit the power of the decoder. This led to better performance even when the induced phrase-tables were noisy, as described below.

The relative performance of the *PhraseTrans* SMT systems compared to the *Direct* SMT systems was 0.92 to 0.97. These are very promising results. To show how these systems translated the test sentences, we translated some outputs of the Spanish-French *Direct* and *PhraseTrans* SMT systems into English using the French-English *Direct* system. These are shown in Table 3 with the reference English sentences.

The relative performance seems to be related to the BLEU scores for the *Direct* SMT systems. It was relatively high (0.95 to 0.97) for the difficult (in terms of BLEU) language pairs but relatively low (0.92) for the easy language pairs; Spanish–French and French–Spanish. There is a lot of room for improvement for the relatively easy language pairs. This relationship is stronger than the relationship between the BLEU scores for SE/ET and those for the *PhraseTrans* systems, where no clear trend exists.

Table 2 shows the number of phrases stored in the phrase-tables. The *Direct* SMT systems had 7.3 to

[8]A typical reranking approach to SMT (Och et al., 2004) uses a 1000–best list.

Source–Target	*Direct*		*PhraseTrans*		*SntTrans15*		*SntTrans1*	SE	ET
Spanish–French	35.78	$>$	32.90 (0.92)	$>$	29.49 (0.82)	$>$	29.16 (0.81)	29.31	28.80
French–Spanish	34.16	$>$	31.49 (0.92)	$>$	28.41 (0.83)	$>$	27.99 (0.82)	27.59	29.07
German–French	23.37	$>$	22.47 (0.96)	$>$	22.03 (0.94)	$>$	21.64 (0.93)	22.40	28.80
French–German	15.27	$>$	14.51 (0.95)	$>$	14.03 (0.92)	$<$	14.21 (0.93)	27.59	15.81
German–Spanish	22.34	$>$	21.76 (0.97)	$>$	21.36 (0.96)	$>$	20.97 (0.94)	22.40	29.07
Spanish–German	15.50	$>$	15.11 (0.97)	$>$	14.46 (0.93)	$<$	14.61 (0.94)	29.31	15.81

Table 1: BLEU scores and relative performance

	No. of phrases ("M" means 10^6)			R	P
	Direct	*PhraseTrans*	common		
S–F	18.2M	190.8M	6.3M	34.7	3.3
F–S	18.2M	186.8M	6.3M	34.7	3.4
G–F	7.3M	174.9M	3.1M	43.2	1.8
F–G	7.3M	168.2M	3.1M	43.2	1.9
G–S	7.5M	179.6M	3.3M	44.1	1.9
S–G	7.6M	176.6M	3.3M	44.1	1.9

"S", "F", and "G" are the acronyms of Spanish, French, and German, respectively. "X–Y" means that "X" is the source language and "Y" is the target language.

Table 2: Statistics for the phrase-tables

18.2 million phrases, and the *PhraseTrans* systems had 168.2 to 190.8 million phrases. The numbers of phrases stored in the *PhraseTrans* systems were very large compared to those of *Direct* systems.[9] However, this does not cause a computational problem in decoding because those phrases that do not appear in source sentences are filtered so that only the relevant phrases are used during decoding.

The figures in the *common* column are the number of phrases common to the *Direct* and *PhraseTrans* systems. R (recall) and P (precision) are defined as follows.

$$\text{R} = \frac{\text{No. of common phrases} \times 100}{\text{No. of phrases in } \textit{Direct} \text{ system}}$$

$$\text{P} = \frac{\text{No. of common phrases} \times 100}{\text{No. of phrases in } \textit{PhraseTrans} \text{ system}}$$

Recall was reasonably high. However, the upper bound of recall was 100 percent because we used a multilingual corpus whose sentences were aligned to each other across all four languages, as described in Section 4.1. Thus, there is a lot of room for improvement with respect to recall. Precision, on the other hand, was very low. However, translation performance was not significantly affected by this low precision, as is shown in Table 1. This indicates that recall is more important than precision in building phrase-tables.

5 Related work

Pivot languages have been used in rule-based machine translation systems. Boitet (1988) discusses the pros and cons of the pivot approaches in multilingual machine translation. Schubert (1988) argues that a pivot language needs to be a natural language, due to the inherent lack of expressiveness of artificial languages.

Pivot-based methods have also been used in other related areas, such as translation lexicon induction (Schafer and Yarowsky, 2002), word alignment (Wang et al., 2006), and cross language information retrieval (Gollins and Sanderson, 2001). The translation disambiguation techniques used in these studies could be used for improving the quality of phrase translation tables.

In contrast to these, very little work has been done on pivot-based methods for SMT. Kauers et al. (2002) used an artificial interlingua for spoken language translation. Gispert and Mariño (2006) created an English-Catalan parallel corpus by automatically translating the Spanish part of an English-Spanish parallel corpus into Catalan with a Spanish-Catalan SMT system. They then directly trained an SMT system on the English-Catalan corpus. They

[9]In Table 2, the *PhraseTrans* systems have more than 10x as many phrases as the *Direct* systems. This can be explained as follows. Let f_i be the *fanout* of an English phrase i, i.e., f_i is the number of phrase pairs containing the English phrase i in a phrase-table, then the size of the phrase-table is $s_1 = \sum_{i=1}^{n} f_i$, where n is the number of distinct English phrases. When we combine two phrase-tables, the size of the combined phrase table is roughly $s_2 = \sum_{i=1}^{n} f_i^2$. Thus, the relative size of the combined phrase table is roughly $r = \frac{s_2}{s_1} = \frac{E(f^2)}{E(f)}$, where $E(f) = \frac{s_1}{n}$ and $E(f^2) = \frac{s_2}{n}$ are the averages over f_i and f_i^2, respectively. As an example, we calculated these averages for the German-English phrase table. $E(f)$ was 1.5, $E(f^2)$ was 43.7, and r was 28.9. This shows that even if an average fanout is small, the size of a combined phrase table can be very large.

showed that this direct training method is superior to the sentence translation strategy (*SntTrans1*) in translating Catalan into English but is inferior to it in the opposite translation direction (in terms of the BLEU score). In contrast, we have shown that the phrase translation strategy consistently outperformed the sentence translation strategy in the controlled experiments.

6 Conclusion

We have compared two types of pivot strategies, namely *phrase translation* and *sentence translation*. The phrase translation strategy directly constructs a phrase translation table from a source language and English phrase-table and a target language and English phrase-table. It then uses this phrase table in a phrase-based SMT system. The sentence translation strategy first translates a source language sentence into n English sentences and translates these n sentences into target language sentences separately. Then, it selects the highest scoring sentence from the target language sentences.

We conducted controlled experiments using the Europarl corpus to compare the performance of these two strategies to that of directly trained SMT systems. The experiments showed that the performance of the phrase translation strategy was statistically significantly better than that of the sentence translation strategy and that its relative performance compared to the directly trained SMT systems was 0.92 to 0.97. These are very promising results.

Although we used the Europarl corpus for controlled experiments, we intend to use the pivot strategies in situations where very limited amount of parallel corpora are available for a source and target language but where relatively large parallel corpora are available for the source language–English and the target language–English. In future work, we will further investigate the pivot strategies described in this paper to confirm that the phrase translation strategy is better than the sentence translation strategy in the intended situation as well as with the Europarl corpus.[10]

[10] As a first step towards real situations, we conducted additional experiments. We divided the training corpora in Section 4 into two halves. We used the first 292915 sentences to train source-English SMT systems and the remaining 292915 ones to train English-target SMT systems. Based on these source-English and English-target SMT systems, we trained *PhraseTrans* and *SntTrans1* SMT systems. Other experimental conditions were the same as those described in Section 4. The table below shows the BLUE scores of these SMT systems. It indicates that the *PhraseTrans* systems consistently outperformed the *SntTrans1* systems.

Source-Target	*PhraseTrans*	*SntTrans1*
Spanish-French	31.57	28.36
French-Spanish	30.18	27.75
German-French	20.48	19.83
French-German	14.38	14.11
German-Spanish	19.58	18.67
Spanish-German	14.80	14.46

References

Christian Boitet. 1988. Pros and cons of the pivot and transfer approaches in multilingual machine translation. In Dan Maxwell, Klaus Schubert, and Toon Witkam, editors, *New Directions in Machine Translation*. Foris. (appeared in Sergei Nirenburg, Harold Somers and Yorick Wilks (eds.) *Readings in Machine Translation* published by the MIT Press in 2003).

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, Philipp Koehn, and Miles Osborne. 2006a. Improved statistical machine translation using paraphrases. In *NAACL*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006b. Re-evaluating the role of BLEU in machine translation research. In *EACL*.

Adriá de Gispert and José B. Mari no. 2006. Catalan-English statistical machine translation without parallel corpus: Bridging through Spanish. In *Proc. of LREC 5th Workshop on Strategies for developing Machine Translation for Minority Languages*.

Tim Gollins and Mark Sanderson. 2001. Improving cross language information retrieval with triangulated translation. In *SIGIR*.

Manuel Kauers, Stephan Vogel, Christian Fügen, and Alex Waibel. 2002. Interlingua based statistical machine translation. In *ICSLP*.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between european languages. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 102–121, New York City, June. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*.

Ref	i hope with all my heart , and i must say this quite emphatically , that an opportunity will arise when this document can be incorporated into the treaties at some point in the future .
Dir	i hope with conviction , and put great emphasis , that again is a serious possibility of including this in the treaties .
PT	i hope with conviction , and i very much , insisted that never be a serious possibility of including this in the treaties .
Ref	should this fail to materialise , we should not be surprised if public opinion proves sceptical about europe , or even rejects it .
Dir	otherwise , we must not be surprised by the scepticism , even the rejection of europe in the public .
PT	otherwise , we must not be surprised by the scepticism , and even the rejection of europe in the public .
Ref	the intergovernmental conference - to address a third subject - on the reform of the european institutions is also of decisive significance for us in parliament .
Dir	the intergovernmental conference - and this i turn to the third issue on the reform of the european institutions is of enormous importance for the european parliament .
PT	the intergovernmental conference - and this brings me to the third issue - on the reform of the european institutions has enormous importance for the european parliament .

Table 3: Reference sentences (Ref) and the English translations (by the French-English *Direct* system) of the outputs of the Spanish-French *Direct* and *PhraseTrans* SMT systems (Dir and PT).

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *IWSLT*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *AMTA*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Makoto Nagao. 1981. A framework of a mechanical translation between Japanese and English by analogy principle. In *the International NATO Symposium on Artificial and Human Intelligence*. (appeared in Sergei Nirenburg, Harold Somers and Yorick Wilks (eds.) *Readings in Machine Translation* published by the MIT Press in 2003).

Sonja Nießen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *EACL*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

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

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*.

Klaus Schubert. 1988. Implicitness as a guiding principle in machine translation. In *COLING*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning Japanese-English news articles and sentences. In *ACL*, pages 72–79.

Haifeng Wang, Hua Wu, and Zhanyi Liu. 2006. Word alignment for languages with scarce resources using bilingual corpora of other language pairs. In *COLING/ACL 2006 Main Conference Poster Sessions*.

Efficient Phrase-table Representation for Machine Translation with Applications to Online MT and Speech Translation

Richard Zens and Hermann Ney

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6 – Computer Science Department
RWTH Aachen University, D-52056 Aachen, Germany
{zens,ney}@cs.rwth-aachen.de

Abstract

In phrase-based statistical machine translation, the phrase-table requires a large amount of memory. We will present an efficient representation with two key properties: *on-demand loading* and a *prefix tree* structure for the source phrases.

We will show that this representation scales well to large data tasks and that we are able to store hundreds of millions of phrase pairs in the phrase-table. For the large Chinese–English NIST task, the memory requirements of the phrase-table are reduced to less than 20 MB using the new representation with no loss in translation quality and speed. Additionally, the new representation is not limited to a specific test set, which is important for online or real-time machine translation.

One problem in speech translation is the matching of phrases in the input word graph and the phrase-table. We will describe a novel algorithm that effectively solves this combinatorial problem exploiting the prefix tree data structure of the phrase-table. This algorithm enables the use of significantly larger input word graphs in a more efficient way resulting in improved translation quality.

1 Introduction

In phrase-based statistical machine translation, a huge number of source and target phrase pairs is memorized in the so-called phrase-table. For medium sized tasks and phrase lengths, these phrase-tables already require several GBs of memory or even do not fit at all. If the source text, which is to be translated, is known in advance, a common trick is to filter the phrase-table and keep a phrase pair only if the source phrase occurs in the text. This filtering is a time-consuming task, as we have to go over the whole phrase-table. Furthermore, we have to repeat this filtering step whenever we want to translate a new source text.

To address these problems, we will use an efficient representation of the phrase-table with two key properties: *on-demand loading* and a *prefix tree* structure for the source phrases. The prefix tree structure exploits the redundancy among the source phrases. Using on-demand loading, we will load only a small fraction of the overall phrase-table into memory. The majority will remain on disk.

The on-demand loading is employed on a per sentence basis, i.e. we load only the phrase pairs that are required for one sentence into memory. Therefore, the memory requirements are low, e.g. less than 20 MB for the Chin.-Eng. NIST task. Another advantage of the on-demand loading is that we are able to translate new source sentences without filtering.

A potential problem is that this on-demand loading might be too slow. To overcome this, we use a binary format which is a memory map of the internal representation used during decoding. Additionally, we load coherent chunks of the tree structure instead of individual phrases, i.e. we have only few disk access operations. In our experiments, the on-demand loading is *not* slower than the traditional approach.

As pointed out in (Mathias and Byrne, 2006), one problem in speech translation is that we have to match the phrases of our phrase-table against a word graph representing the alternative ASR transcriptions. We will present a phrase matching algorithm that effectively solves this combinatorial problem exploiting the prefix tree data structure of the phrase-table. This algorithm enables the use of significantly larger input word graphs in a more efficient way resulting in improved translation quality.

The remaining part is structured as follows: we will first discuss related work in Sec. 2. Then, in Sec. 3, we will describe the phrase-table representation. Afterwards, we will present applications in speech translation and online MT in Sec. 4 and 5, respectively. Experimental results will be presented in Sec. 6 followed by the conclusions in Sec. 7.

2 Related Work

(Callison-Burch et al., 2005) and (Zhang and Vogel, 2005) presented data structures for a compact representation of the word-aligned bilingual data, such that on-the-fly extraction of long phrases is possible. The motivation in (Callison-Burch et al., 2005) is that there are some long source phrases in the test data that also occur in the training data. However, the more interesting question is if these long phrases really help to improve the translation quality. We have investigated this and our results are in line with (Koehn et al., 2003) showing that the translation quality does *not* improve if we utilize phrases beyond a certain length. Furthermore, the suffix array data structure of (Callison-Burch et al., 2005) requires a fair amount of memory, about 2 GB in their example, whereas our implementation will use only a tiny amount of memory, e.g. less than 20 MB for the large Chinese-English NIST task.

3 Efficient Phrase-table Representation

In this section, we will describe the proposed representation of the phrase-table. A prefix tree, also called trie, is an ordered tree data structure used to store an associative array where the keys are symbol sequences. In the case of phrase-based MT, the keys are source phrases, i.e. sequences of source words and the associated values are the possible translations of these source phrases. In a prefix tree, all descendants of any node have a common prefix, namely the source phrase associated with that node. The root node is associated with the empty phrase.

The prefix tree data structure is quite common in automatic speech translation. There, the lexicon, i.e. the mapping of phoneme sequences to words, is usually organized as a prefix tree (Ney et al., 1992).

We convert the list of source phrases into a prefix tree and, thus, exploit that many of them share the same prefix. This is illustrated in Fig. 1 (left). Within each node of the tree, we store a sorted array of possible successor words along with pointers to the corresponding successor nodes. Additionally, we store a pointer to the possible translations.

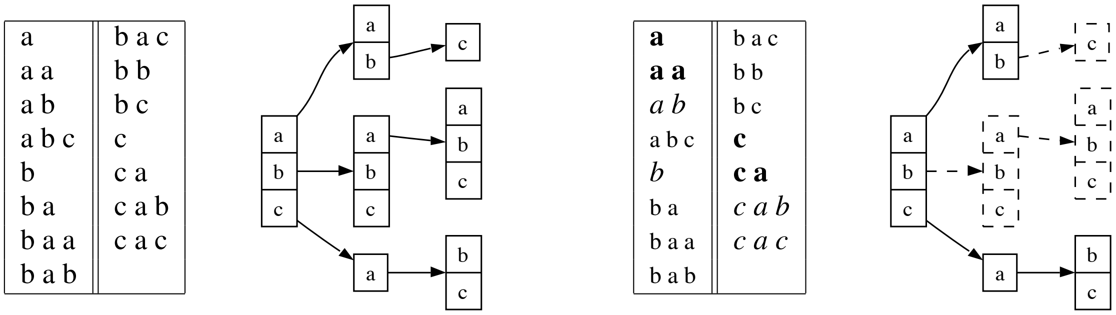
One property of the tree structure is that we can efficiently access the successor words of a given prefix. This will be a key point to achieve an efficient phrase matching algorithm in Sec. 4. When looking for a specific successor word, we perform a binary search in the sorted array. Alternatively, we could use hashing to speed up this lookup. We have chosen an array representation as this can be read very fast from disk. Additionally, with the exception of the root node, the branching factor of the tree is small, i.e. the potential benefit from hashing is limited. At the root node, however, the branching factor is close to the vocabulary size of the source language, which can be large. As we store the words internally as integers and virtually all words occur as the first word of some phrase, we can use the integers directly as the position in the array of the root node. Hence, the search for the successors at the root node is a simple table lookup with direct access, i.e. in $\mathcal{O}(1)$.

If not filtered for a specific test set, the phrase-table becomes huge even for medium-sized tasks. Therefore, we store the tree structure on disk and load only the required parts into memory on-demand. This is illustrated in Fig. 1 (right). Here, we show the matching phrases for the source sentence 'c a a c', where the matching phrases are set in **bold** and the phrases that are loaded into memory are set in *italics*. The dashed part of the tree structure is not loaded into memory. Note that some nodes of the tree are loaded even if there is no matching phrase in that node. These are required to actually verify that there is no matching phrase. An example is the 'bc' node in the lower right part of the figure. This node is loaded to check if the phrase 'c a a' occurs in the phrase-table. The translations, however, are loaded only for matching source phrases.

In the following sections, we will describe applications of this phrase-table representation for speech translation and online MT.

4 Speech Translation

In speech translation, the input to the MT system is not a sentence, but a word graph representing alter-



a	b a c
a a	b b
a b	b c
a b c	c
b	c a
b a	c a b
b a a	c a c
b a b	

a	b a c
a a	b b
a b	b c
a b c	**c**
b	**c a**
b a	*c a b*
b a a	*c a c*
b a b	

Figure 1: Illustration of the prefix tree. Left: list of source phrases and the corresponding prefix tree. Right: list of matching source phrases for sentence 'c a a c' (**bold** phrases match, phrases in *italics* are loaded in memory) and the corresponding partially loaded prefix tree (the dashed part is not in memory).

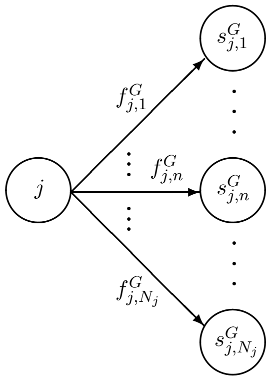


Figure 2: Illustration for graph G: node j with successor nodes $s^G_{j,1}, ..., s^G_{j,n} ..., s^G_{j,N_j}$ and corresponding edge labels $f^G_{j,1}, ..., f^G_{j,n}, ..., f^G_{j,N_j}$.

native ASR transcriptions. As pointed out in (Mathias and Byrne, 2006), one problem in speech translation is that we have to match the phrases of our phrase-table against the input word graph. This results in a combinatorial problem as the number of phrases in a word graph increases exponentially with the phrase length.

4.1 Problem Definition

In this section, we will introduce the notation and state the problem of matching source phrases of an input graph G and the phrase-table, represented as prefix tree T. The input graph G has nodes $1, ..., j, ..., J$. The outgoing edges of a graph node j are numbered with $1, ..., n, ..., N_j$, i.e. an edge in the input graph is identified by a pair (j, n). The source word labeling the n^{th} outgoing edge of graph node j is denoted as $f^G_{j,n}$ and the successor node of this edge is denoted as $s^G_{j,n} \in \{1, ..., J\}$. This notation is illustrated in Fig. 2.

We use a similar notation for the prefix tree T with nodes $1, ..., k, ..., K$. The outgoing edges of a tree node k are numbered with $1, ..., m, ..., M_k$, i.e. an edge in the prefix tree is identified by a pair (k, m). The source word labeling the m^{th} outgoing edge of tree node k is denoted as $f^T_{k,m}$ and the successor node of this edge is denoted as $s^T_{k,m} \in \{1, ..., K\}$. Due to the tree structure, the successor nodes of a tree node k are all distinct:

$$s^T_{k,m} = s^T_{k,m'} \quad \Leftrightarrow \quad m = m' \tag{1}$$

Let k_0 denote the root node of the prefix tree and let $\tilde{f}_k$ denote the prefix that leads to tree node k. Furthermore, we define $E(k)$ as the set of possible translations of the source phrase $\tilde{f}_k$. These are the entries of the phrase-table, i.e.

$$E(k) = \left\{ \tilde{e} \,\middle|\, p(\tilde{e}|\tilde{f}_k) > 0 \right\} \tag{2}$$

We will need similar symbols for the input graph. Therefore, we define $F(j', j)$ as the set of source phrases of all paths from graph node j' to node j, or formally:

$$F(j', j) = \left\{ \tilde{f} \,\middle|\, \exists (j_i, n_i)_{i=1}^I : \tilde{f} = f^G_{j_1,n_1}, ..., f^G_{j_I,n_I} \right. \\ \left. \wedge j_1 = j' \wedge \bigwedge_{i=1}^{I-1} s^G_{j_i,n_i} = j_{i+1} \wedge s_{j_I,n_I} = j \right\}$$

Here, the conditions ensure that the edge sequence $(j_i, n_i)_{i=1}^I$ is a proper path from node j' to node j in the input graph and that the corresponding source phrase is $\tilde{f} = f^G_{j_1,n_1}, ..., f^G_{j_I,n_I}$. This definition can be expressed in a recursive way; the idea is to extend the phrases of the predecessor nodes by one word:

$$F(j', j) = \bigcup_{(j'',n):s^G_{j'',n}=j} \left\{ \tilde{f} f^G_{j'',n} \,\middle|\, \tilde{f} \in F(j', j'') \right\} \tag{3}$$

Here, the set is expressed as a union over all inbound edges (j'', n) of node j. We concatenate each source phrase $\tilde{f}$ that ends at the start node of such an edge, i.e. $\tilde{f} \in F(j', j'')$, with the corresponding edge label $f^G_{j'',n}$. Additionally, we define $E(j', j)$ as the set of possible translations of all paths from graph node j' to graph node j, or formally:

$$E(j', j) = \left\{ \tilde{e} \,\middle|\, \exists \tilde{f} \in F(j', j) : p(\tilde{e}|\tilde{f}) > 0 \right\} \quad (4)$$

$$= \bigcup_{k:\tilde{f}_k \in F(j',j)} E(k) \quad (5)$$

$$= \bigcup_{(j'',n):s^G_{j'',n}=j} \quad \bigcup_{\substack{k:\tilde{f}_k \in F(j',j'') \\ m:f^G_{j'',n}=f^T_{k,m}}} E(s^T_{k,m}) \quad (6)$$

Here, the definition was first rewritten using Eq. 2 and then using Eq. 3. Again, the set is expressed recursively as a union over the inbound edges. For each inbound edge (j'', n), the inner union verifies that there exists a corresponding edge (k, m) in the prefix tree with the same label, i.e. $f^G_{j'',n} = f^T_{k,m}$.

Our goal is to find all non-empty sets of translation options $E(j', j)$. The naive approach would be to enumerate all paths in the input graph from node j' to node j, then lookup the corresponding source phrase in the phrase-table and add the translations, if there are any, to the set of translation options $E(j', j)$. This solution has some obvious weaknesses: the number of paths between two nodes is typically huge and the majority of the corresponding source phrases do not occur in the phrase-table.

We omitted the probabilities for notational convenience. The extensions are straightforward. Note that we store only the target phrases $\tilde{e}$ in the set of possible translations $E(j', j)$ and not the source phrases $\tilde{f}$. This is based on the assumption that the models which are conditioned on the source phrase $\tilde{f}$ are independent of the context outside the phrase pair $(\tilde{f}, \tilde{e})$. This assumption holds for the standard phrase and word translation models. Thus, we have to keep only the target phrase with the highest probability. It might be violated by lexicalized distortion models (dependent on the configuration); in that case we have to store the source phrase along with the target phrase and the probability, which is again straightforward.

4.2 Algorithm

The algorithm for matching the source phrases of the input graph G and the prefix tree T is presented in Fig. 3. Starting from a graph node j', we explore the part of the graph which corresponds to known source phrase prefixes and generate the sets $E(j', j)$ incrementally based on Eq. 6. The intermediate states are represented as pairs (j, k) meaning that there exists a path in the input graph from node j' to node j which is labeled with the source phrase $\tilde{f}_k$, i.e. the source phrase that leads to node k in the prefix tree. These intermediate states are stored on a stack. After the initialization in line 1, the main loop starts. We take one item from the stack and update the translation options $E(j', j)$ in line 4. Then, we loop over all outgoing edges of the current graph node j. For each edge, we first check if the edge is labeled with an ϵ in line 6. In this special case, we go to the successor node in the input graph $s^G_{j,n}$, but remain in the current node k of the prefix tree. In the regular case, i.e. the graph edge label is a regular word, we check in line 8 if the current prefix tree node k has an outgoing edge labeled with that word. If such an edge is found, we put a new state on the stack with the two successor nodes in the input graph $s^G_{j,n}$ and the prefix tree $s^T_{k,m}$, respectively.

Figure 3: Algorithm `phrase-match` for matching source phrases of input graph G and prefix tree T. Input: graph G, prefix tree T, translation options $E(k)$ for all tree nodes k; output: translation options $E(j', j)$ for all graph nodes j' and j.

```
0 FOR $j' = 1$ TO $J$ DO
1     stack.push($j', k_0$)
2     WHILE not stack.empty() DO
3         ($j, k$) = stack.pop()
4         $E(j', j) = E(j', j) \cup E(k)$
5         FOR $n = 1$ TO $N_j$ DO
6             IF ($f^G_{j,n} = \epsilon$)
7             THEN stack.push($s^G_{j,n}, k$)
8             ELSE IF ($\exists m : f^G_{j,n} = f^T_{k,m}$)
9                 THEN stack.push($s^G_{j,n}, s^T_{k,m}$)
```

4.3 Computational Complexity

In this section, we will analyze the computational complexity of the algorithm. The computational complexity of lines 5-9 is in $\mathcal{O}(N_j \log M_k)$, i.e. it depends on the branching factors of the input graph and the prefix tree. Both are typically small. An exception is the branching factor of the root node k_0 of the prefix tree, which can be rather large, typically it is the vocabulary size of the source language. But, as described in Sec. 3, we can access the successor nodes of the root node of the prefix tree in $\mathcal{O}(1)$, i.e. in constant time. So, if we are at the root node of the prefix tree, the computational complexity of lines 5-9 is in $\mathcal{O}(N_j)$. Using hashing at the interior nodes of the prefix tree would result in a constant time lookup at these nodes as well. Nevertheless, the sorted array implementation that we chose has the advantage of faster loading from disk which seems to be more important in practice.

An alternative interpretation of lines 5-9 is that we have to compute the intersection of the two sets f_j^G and f_k^T, with

$$f_j^G = \{f_{j,n}^G \mid n = 1, ..., N_j\} \quad (7)$$
$$f_k^T = \{f_{k,m}^T \mid m = 1, ..., M_k\}. \quad (8)$$

Assuming both sets are sorted, this could be done in linear time, i.e. in $\mathcal{O}(N_j + M_k)$. In our case, only the edges in the prefix tree are sorted. Obviously, we could sort the edges in the input graph and then apply the linear algorithm, resulting in an overall complexity of $\mathcal{O}(N_j \log N_j + M_k)$. As the algorithm visits nodes multiple times, we could do even better by sorting all edges of the graph during the initialization. Then, we could always apply the linear time method. On the other hand, it is unclear if this pays off in practice and an experimental comparison has to be done which we will leave for future work.

The overall complexity of the algorithm depends on how many phrases of the input graph occur in the phrase-table. In the worst case, i.e. if all phrases occur in the phrase-table, the described algorithm is not more efficient than the naive algorithm which simply enumerates all phrases. Nevertheless, this does not happen in practice and we observe an exponential speed up compared to the naive algorithm, as will be shown in Sec. 6.3.

5 Online Machine Translation

Beside speech translation, the presented phrase-table data structure has other interesting applications. One of them is online MT, i.e. an MT system that is able to translate unseen sentences without significant delay. These online MT systems are typically required if there is some interaction with human users, e.g. if the MT system acts as an interpreter in a conversation, or in real-time systems. This situation is different from the usual research environment where typically a fair amount of time is spent to prepare the MT system to translate a certain set of source sentences. In the research scenario, this preparation usually pays off as the same set of sentences is translated multiple times. In contrast, an online MT system translates each sentence just once. One of the more time-consuming parts of this preparation is the filtering of the phrase-table. Using the on-demand loading technique we described in Sec. 3, we can avoid the filtering step and directly translate the source sentence. An additional advantage is that we load only small parts of the full phrase-table into memory. This reduces the memory requirements significantly, e.g. for the Chinese–English NIST task, the memory requirement of the phrase-table is reduced to less than 20 MB using on-demand loading. This makes the MT system usable on devices with limited hardware resources.

Table 1: NIST task: corpus statistics.

			Chinese	English
Train	Sentence pairs		7 M	
	Running words		199 M	213 M
	Vocabulary size		222 K	351 K
Test	2002	Sentences	878	3 512
		Running words	25 K	105 K
	2005	Sentences	1 082	4 328
		Running words	33 K	148 K

6 Experimental Results

6.1 Translation System

For the experiments, we use a state-of-the-art phrase-based statistical machine translation system as described in (Zens and Ney, 2004). We use a log-linear combination of several models: a four-gram language model, phrase-based and word-based translation models, word, phrase and distortion penalty and a lexicalized distortion model. The model scaling factors are optimized using minimum error rate training (Och, 2003).

6.2 Empirical Analysis for a Large Data Task

In this section, we present an empirical analysis of the described data structure for the large data track of the Chinese-English NIST task. The corpus statistics are shown in Tab. 1.

The translation quality is measured using two accuracy measures: the BLEU and the NIST score. Additionally, we use the two error rates: the word error rate (WER) and the position-independent word error rate (PER). These evaluation criteria are computed with respect to four reference translations.

In Tab. 2, we present the translation quality as a

Table 2: NIST task: translation quality as a function of the maximum source phrase length.

src	NIST 2002 set (dev)				NIST 2005 set (test)			
len	WER[%]	PER[%]	BLEU[%]	NIST	WER[%]	PER[%]	BLEU[%]	NIST
1	71.9	46.8	27.07	8.37	78.0	49.0	23.11	7.62
2	62.4	41.2	34.36	9.39	68.5	42.2	30.32	8.74
3	62.0	41.1	34.89	9.33	67.7	42.1	30.90	8.74
4	61.7	41.1	35.05	9.27	67.6	41.9	30.99	8.75
5	61.8	41.2	34.95	9.25	67.6	41.9	30.93	8.72
∞	61.8	41.2	34.99	9.25	67.5	41.8	30.90	8.73

Table 3: NIST task: phrase-table statistics.

src	number of distinct		avg. tgt
len	src phrases	src-tgt pairs	candidates
1	221 505	17 456 415	78.8
2	5 000 041	39 436 617	7.9
3	20 649 699	58 503 904	2.8
4	31 383 549	58 436 271	1.9
5	32 679 145	51 255 866	1.6
total	89 933 939	225 089 073	2.5

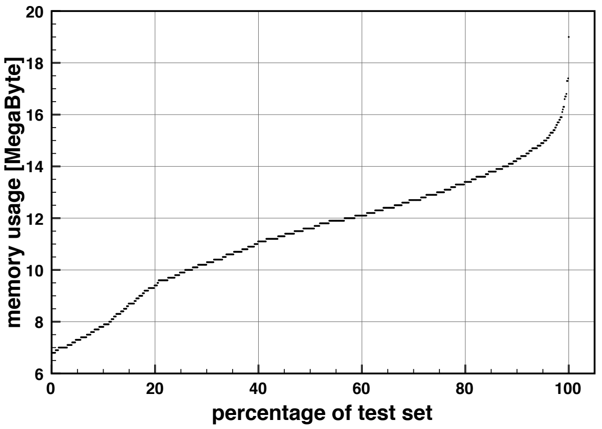


Figure 4: NIST task: phrase-table memory usage per sentence (sorted).

function of the maximum source phrase length. We observe a large improvement when going beyond length 1, but this flattens out very fast. Using phrases of lengths larger than 4 or 5 does *not* result in further improvement. Note that the minor differences in the evaluation results for length 4 and beyond are merely statistical noise. Even a length limit of 3, as proposed by (Koehn et al., 2003), would result in almost optimal translation quality. In the following experiments on this task, we will use a limit of 5 for the source phrase length.

In Tab. 3, we present statistics about the extracted phrase pairs for the Chinese–English NIST task as a function of the source phrase length, in this case for length 1-5. The phrases are not limited to a specific test set. We show the number of distinct source phrases, the number of distinct source-target phrase pairs and the average number of target phrases (or translation candidates) per source phrase. In the experiments, we limit the number of translation candidates per source phrase to 200. We store a total of almost 90 million distinct source phrases and more than 225 million distinct source-target phrase pairs in the described data structure. Obviously, it would be infeasible to load this huge phrase-table completely into memory. Nevertheless, using on-demand loading, we are able to utilize all these phrase pairs with minimal memory usage.

In Fig. 4, we show the memory usage of the described phrase-table data structure per sentence for the NIST 2002 test set. The sentences were sorted according to the memory usage. The maximum amount of memory for the phrase-table is 19 MB; for more than 95% of the sentences no more than 15 MB are required. Storing all phrase pairs for this test set in memory requires about 1.7 GB of memory, i.e. using the described data structures, we not only avoid the limitation to a specific test set, but we also reduce the memory requirements by about two orders of a magnitude.

Another important aspect that should be considered is translation speed. In our experiments, the described data structure is *not* slower than the traditional approach. We attribute this to the fact that we use a binary format that is a memory map of the data structure used internally and that we load the data in rather large, coherent chunks. Additionally, there is virtually no initialization time for the phrase-table which decreases the overhead of parallelization and therefore speeds up the development cycle.

6.3 Speech Translation

The experiments for speech translation were conducted on the European Parliament Plenary Sessions (EPPS) task. This is a Spanish-English speech-to-speech translation task collected within the TC-Star

Table 4: EPPS task: corpus statistics.

Train	Spanish	English
Sentence pairs	1.2 M	
Running words	31 M	30 M
Vocabulary size	140 K	94 K
Test confusion networks	**Full**	**Pruned**
Sentences	1 071	
Avg. length	23.6	
Avg. / max. depth	2.7 / 136	1.3 / 11
Avg. number of paths	10^{75}	264 K

project. The training corpus statistics are presented in Tab. 4. The phrase-tables for this task were kindly provided by ITC-IRST.

We evaluate the `phrase-match` algorithm in the context of confusion network (CN) decoding (Bertoldi and Federico, 2005), which is one approach to speech translation. CNs (Mangu et al., 2000) are interesting for MT because the reordering can be done similar to single best input. For more details on CN decoding, please refer to (Bertoldi et al., 2007). Note that the `phrase-match` algorithm is not limited to CNs, but can work on arbitrary word graphs.

Statistics of the CNs are also presented in Tab. 4. We distinguish between the full CNs and pruned CNs. The pruning parameters were chosen such that the resulting CNs are similar in size to the largest ones in (Bertoldi and Federico, 2005). The average depth of the full CNs, i.e. the average number of alternatives per position, is about 2.7 words whereas the maximum is as high as 136 alternatives.

In Fig. 5, we present the average number of phrase-table look-ups for the full EPPS CNs as a function of the source phrase length. The curve 'CN total' represents the total number of source phrases in the CNs for a given length. This is the number of phrase-table look-ups using the naive algorithm. Note the exponential growth with increasing phrase length. Therefore, the naive algorithm is only applicable for very short phrases and heavily pruned CNs, as e.g. in (Bertoldi and Federico, 2005).

The curve 'CN explored' is the number of phrase-table look-ups using the `phrase-match` algorithm described in Fig. 3. We do *not* observe the exponential explosion as for the naive algorithm. Thus, the presented algorithm effectively solves the combinatorial problem of matching phrases of the input CNs and the phrase-table. For comparison, we plotted also the number of look-ups using the `phrase-match` algorithm in the case of single-best input, labeled 'single-best explored'. The maximum phrase length for these experiments is seven. For CN input, this length can be exceeded as the CNs may contain ϵ-transitions.

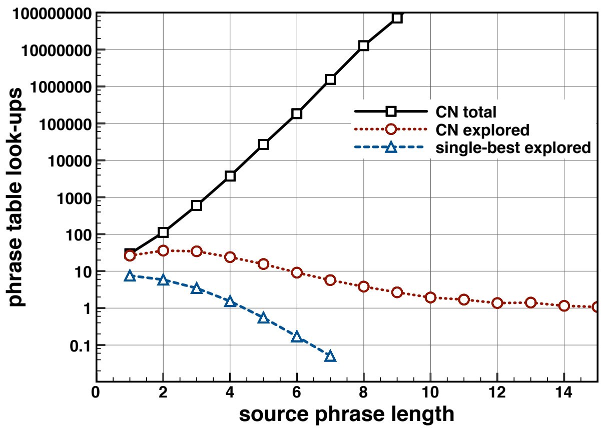


Figure 5: EPPS task: avg. number of phrase-table look-ups per sentence as a function of the source phrase length.

Table 5: EPPS task: translation quality and time for different input conditions (CN=confusion network, time in seconds per sentence).

Input type	BLEU[%]	Time [sec]
Single best	37.6	2.7
CN pruned	38.5	4.8
full	38.9	9.2

In Tab. 5, we present the translation results and the translation times for different input conditions. We observe a significant improvement in translation quality as more ASR alternatives are taken into account. The best results are achieved for the full CNs. On the other hand, the decoding time increases only moderately. Using the new algorithm, the ratio of the time for decoding the CNs and the time for decoding the single best input is 3.4 for the full CNs and 1.8 for the pruned CNs. In previous work (Bertoldi and Federico, 2005), the ratio for the pruned CNs was about 25 and the full CNs could not be handled.

To summarize, the presented algorithm has two main advantages for speech translation: first, it enables us to utilize large CNs, which was prohibitively expensive beforehand and second, the efficiency is improved significantly.

Whereas the previous approaches required careful pruning of the CNs, we are able to utilize the unpruned CNs. Experiments on other tasks have shown that even larger CNs are unproblematic.

7 Conclusions

We proposed an efficient phrase-table data structure which has two key properties:

1. **On-demand loading.**
   We are able to store hundreds of millions of phrase pairs and require only a very small amount of memory during decoding, e.g. less than 20 MB for the Chinese-English NIST task. This enables us to run the MT system on devices with limited hardware resources or alternatively to utilize the freed memory for other models. Additionally, the usual phrase-table filtering is obsolete, which is important for online MT systems.

2. **Prefix tree data structure.**
   Utilizing the prefix tree structure enables us to efficiently match source phrases against the phrase-table. This is especially important for speech translation where the input is a graph representing a huge number of alternative sentences. Using the novel algorithm, we are able to handle large CNs, which was prohibitively expensive beforehand. This results in more efficient decoding and improved translation quality.

We have shown that this data structure scales very well to large data tasks like the Chinese-English NIST task. The implementation of the described data structure as well as the `phrase-match` algorithm for confusion networks is available as open source software in the MOSES toolkit[1].

Not only standard phrase-based systems can benefit from this data structure. It should be rather straightforward to apply this data structure as well as the `phrase-match` algorithm to the hierarchical approach of (Chiang, 2005). As the number of rules in this approach is typically larger than the number of phrases in a standard phrase-based system, the gains should be even larger.

The language model is another model with high memory requirements. It would be interesting to investigate if the described techniques and data structures are applicable for reducing the memory requirements of language models.

Some aspects of the `phrase-match` algorithm are similar to the composition of finite-state automata. An efficient implementation of on-demand loading (not only on-demand computation) for a finite-state toolkit would make the whole range of finite-state operations applicable to large data tasks.

[1]http://www.statmt.org/moses

Acknowledgments

This material is partly based upon work supported by the DARPA under Contract No. HR0011-06-C-0023, and was partly funded by the European Union under the integrated project TC-STAR (IST-2002-FP6-506738, http://www.tc-star.org). Additionally, we would like to thank all group members of the JHU 2006 summer research workshop *Open Source Toolkit for Statistical Machine Translation*.

References

N. Bertoldi and M. Federico. 2005. A new decoder for spoken language translation based on confusion networks. In Proc. *IEEE Automatic Speech Recognition and Understanding Workshop*, pages 86–91, Mexico, November/December.

N. Bertoldi, R. Zens, and M. Federico. 2007. Speech translation by confusion networks decoding. In Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, April.

C. Callison-Burch, C. Bannard, and J. Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 255–262, Ann Arbor, MI, June.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 263–270, Ann Arbor, MI, June.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pages 127–133, Edmonton, Canada, May/June.

L. Mangu, E. Brill, and A. Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer, Speech and Language*, 14(4):373–400, October.

L. Mathias and W. Byrne. 2006. Statistical phrase-based speech translation. In Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 561–564, Toulouse, France, May.

H. Ney, R. Haeb-Umbach, B. H. Tran, and M. Oerder. 1992. Improvements in beam search for 10000-word continuous speech recognition. In Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 9–12, San Francisco, CA, March.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In Proc. *41st Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pages 257–264, Boston, MA, May.

Y. Zhang and S. Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrases and large corpora. In Proc. *10th Annual Conf. of the European Assoc. for Machine Translation (EAMT)*, pages 294–301, Budapest, Hungary, May.

An Efficient Two-Pass Approach to Synchronous-CFG Driven Statistical MT

Ashish Venugopal and **Andreas Zollmann** and **Stephan Vogel**
School of Computer Science, Carnegie Mellon University, Pittsburgh
interACT Lab, University of Karlsruhe
{ashishv,zollmann,vogel+}@cs.cmu.edu

Abstract

We present an efficient, novel two-pass approach to mitigate the computational impact resulting from online intersection of an n-gram language model (LM) and a probabilistic synchronous context-free grammar (PSCFG) for statistical machine translation. In first pass CYK-style decoding, we consider first-best chart item approximations, generating a hypergraph of sentence spanning target language derivations. In the second stage, we instantiate specific alternative derivations from this hypergraph, using the LM to *drive* this search process, recovering from search errors made in the first pass. Model search errors in our approach are comparable to those made by the state-of-the-art "Cube Pruning" approach in (Chiang, 2007) under comparable pruning conditions evaluated on both hierarchical and syntax-based grammars.

1 Introduction

Syntax-driven (Galley et al., 2006) and hierarchical translation models (Chiang, 2005) take advantage of probabilistic synchronous context free grammars (PSCFGs) to represent structured, lexical reordering constraints during the decoding process. These models extend the domain of locality (over phrase-based models) during decoding, representing a significantly larger search space of possible translation derivations. While PSCFG models are often induced with the goal of producing grammatically correct target translations as an implicit syntax-structured language model, we acknowledge the value of n-gram language models (LM) in phrase-based approaches.

Integrating n-gram LMs into PSCFGs based decoding can be viewed as online intersection of the PSCFG grammar with the finite state machine represented by the n-gram LM, dramatically increasing the effective number of nonterminals in the decoding grammar, rendering the decoding process essentially infeasible without severe, beam-based lossy pruning. The alternative, simply decoding without the n-gram LM and rescoring N-best alternative translations, results in substantially more search errors, as shown in (Zollmann and Venugopal, 2006).

Our two-pass approach involves fast, approximate synchronous parsing in a first stage, followed by a second, detailed exploration through the resulting hypergraph of sentence spanning derivations, using the n-gram LM to drive that search. This achieves search errors comparable to a strong "Cube Pruning" (Chiang, 2007), single-pass baseline. The first pass corresponds to a severe parameterization of Cube Pruning considering only the first-best (LM integrated) chart item in each cell while maintaining unexplored alternatives for second-pass consideration. Our second stage allows the integration of long distance and flexible history n-gram LMs to drive the search process, rather than simply using such models for hypothesis rescoring.

We begin by discussing the PSCFG model for statistical machine translation, motivating the need for effective n-gram LM integration during decoding. We then present our two-pass approach and discuss Cube Pruning as a state-of-the-art baseline. We present results in the form of search error analysis and translation quality as measured by the BLEU score (Papineni et al., 2002) on the IWSLT 06 text translation task (Eck and Hori, 2005)[1], comparing Cube Pruning with our two-pass approach.

2 Synchronous Parsing for SMT

Probabilistic Synchronous Context Free Grammar (PSCFG) approaches to statistical machine translation use a source terminal set (source vocabulary) $\mathcal{T}_S$, a target terminal set (target vocabulary) $\mathcal{T}_T$ and a shared nonterminal set $\mathcal{N}$ and induce rules of the form

$$X \to \langle \gamma, \alpha, \sim, w \rangle$$

where (i) $X \in \mathcal{N}$ is a nonterminal, (ii) $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)^*$ is a sequence of nonterminals and source terminals, (iii) $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)^*$ is a sequence of nonterminals and target terminals, (iv) the number $cnt(\gamma)$ of nonterminal occurrences in γ is equal to the number $cnt(\alpha)$ of nonterminal occurrences in α, (v) $\sim$: $\{1, \ldots, cnt(\gamma)\} \to \{1, \ldots, cnt(\alpha)\}$ is a one-to-one mapping from nonterminal occurrences in γ to nonterminal occurrences in α, and (vi) $w \in [0, \infty)$ is a non-negative real-valued weight assigned to the rule. We will assume $\sim$ to be implicitly defined by indexing the NT occurrences in γ from left to right starting with 1, and by indexing the NT occurrences in α by the indices of their corresponding counterparts in γ. Syntax-oriented PSCFG approaches typically ignore source structure, instead focussing on generating syntactically well formed target derivations. (Galley et al., 2006) use syntactic constituents for the PSCFG nonterminal set and (Zollmann and Venugopal, 2006) take advantage of CCG (Steedman, 1999) categories, while (Chiang, 2005) uses a single generic nonterminal. PSCFG derivations function analogously to CFG derivations. Given a source sentence f, the translation task under a PSCFG grammar can be expressed as

[1]While IWSLT represents a limited resource translation task (120K sentences of training data for Chinese-English), the problem of efficient n-gram LM integration is still critically important to efficient decoding, and our contributions can be expected to have an even more significant impact when decoding with grammars induced from larger corpora.

$$\hat{e} = \underset{\{e \mid \exists D.\ \mathrm{src}(D)=f, \mathrm{tgt}(D)=e\}}{\arg\max} P(D)$$

where $\mathrm{tgt}(D)$ refers to the target terminal symbols generated by the derivation D and $\mathrm{src}(D)$ refers to the source terminal symbols spanned by D. The score (also laxly called probability, since we never need to compute the partition function) of a derivation D under a log-linear model, referring to the rules r used in D, is:

$$P(D) = \frac{1}{Z} P_{LM}(\mathrm{tgt}(D))^{\lambda_{LM}} \times \prod_i \prod_{r \in D} \phi_i(r)^{\lambda_i}$$

where ϕ_i refers to features defined on each rule, and P_{LM} is a g-gram LM probability applied to the target terminal symbols generated by the derivation D. Introducing the LM feature defines dependencies across adjacent rules used in each derivation, and requires modifications to the decoding strategy. Viewing the LM as a finite-state machine, the decoding process involves performing an intersection between the PSCFG grammar and the g-gram LM (Bar-Hillel et al., 1964). We present our work under the construction in (Wu, 1996), following notation from (Chiang, 2007), extending the formal description to reflect grammars with an arbitrary number of nonterminals in each rule.

2.1 Decoding Strategies

In Figure 1, we reproduce the decoding algorithm from (Chiang, 2007) that applies a PSCFG to translate a source sentence in the same notation (as a deductive proof system (Shieber et al., 1995)), generalized to handle more than two non-terminal pairs. Chart items $[X, i, j, e] : w$ span $j - i$ words in the source sentence $f_1 \cdots f_n$, starting at position $i + 1$, and have weight w (equivalent to $P(D)$), and $e \in (\mathcal{T}_T \cup \{\star\})^*$ is a sequence of target terminals, with possible elided parts, marked by $\star$. Functions p, q whose domain is $\mathcal{T}_T \cup \{\star\}$ are defined in (Chiang, 2007) and are repeated here for clarity.

$$p(a_1 \cdots a_m) = \prod_{g \le i \le m, \star \notin a_{i-g+1} \cdots a_{i-1}} P_{LM}(a_i | a_{i-g+1} \cdots a_{i-1})$$

$$q(a_1 \cdots a_m) = \begin{cases} a_1 \cdots a_{g-1} \star a_{m-g+2} \cdots a_m & \text{if } m \ge g \\ a_1 \cdots a_m & \text{else} \end{cases}$$

The function q elides elements from a target language terminal sequence, leaving the leftmost and rightmost $g - 1$ words, replacing the elided words

$$\frac{}{X \to \langle \gamma, \alpha \rangle : w} \quad (X \to \langle \gamma, \alpha, w \rangle) \in G$$

$$\frac{X \to \langle f_{i+1}^{j}, \alpha \rangle : w}{[X, i, j; q(\alpha)] : wp(\alpha)}$$

$$\frac{Z \to \langle f_{i+1}^{i_1} (X^1)_1 f_{j_1+1}^{i_2} \cdots (X^{m-1})_{m-1} f_{j_{m-1}+1}^{i_m} (X^m)_m f_{j_m+1}^{j}, \alpha \rangle : w \quad \left[X^1, i_1, j_1; e_1\right] : w_1 \ \cdots \ \left[X^m, i_m, j_m; e_m\right] : w_m}{[Z, i, j, q(\alpha')] : ww_1 \cdots w_m p(\alpha') \quad (\text{where } \alpha' = \alpha \left[e_1/(X^1)_1, \ldots, e_m/(X^m)_m\right])}$$

$$\text{Goal item: } \left[S, 0, n; \langle s \rangle^{g-1} \star \langle \backslash s \rangle^{g-1}\right]$$

Figure 1. CYK parsing with integrated g-gram LM. The inference rules are explored in ascending order of $j - i$. Here $\alpha\,[e/Y_i]$ is the string α where the NT occurrence Y_i is replaced by e. Functions q and p are explained in the text.

with a single $\star$ symbol. The function p returns g-gram LM probabilities for target terminal sequences, where the $\star$ delineates context boundaries, preventing the calculation from spanning this boundary. We add a distinguished start nonterminal S to generate sentences spanning target translations beginning with $g-1$ $\langle s \rangle$ symbols and ending with $g-1$ $\langle \backslash s \rangle$ symbols. This can e.g. be achieved by adding for each nonterminal X a PSCFG rule

$$S \to \langle X, \langle s \rangle^{g-1} X \langle \backslash s \rangle^{g-1}, 1 \rangle$$

We are only searching for the derivation of highest probability, so we can discard identical chart items that have lower weight. Since chart items are defined by their left-hand side nonterminal production, span, *and* the LM contexts e, we can safely discard these identical items since q has retained all context that could possibly impact the LM calculation. This process is commonly referred to as item recombination. Backpointers to antecedent cells are typically retained to allow N-Best extraction using an algorithm such as (Huang and Chiang, 2005).

The impact of g-gram LM intersection during decoding is apparent in the final deduction step. Generating the set of consequent Z chart items involves combining m previously produced chart cells. Since each of these chart cells with given source span $[i, j]$ is identified by nonterminal symbol X and LM context e, we have at worst $|\mathcal{N}| * |\mathcal{T}_T|^{2(g-1)}$ such chart cells in a span. The runtime of this algorithm is thus

$$\mathcal{O}\left(n^3 \left[|\mathcal{N}||\mathcal{T}_T|^{2(g-1)}\right]^K\right)$$

where K is the maximum number of NT pairs per rule and n the source sentence length. Without severe pruning, this runtime is prohibitive for even the smallest induced grammars. Traditional pruning approaches that limit the number of consequents after they are produced are not effective since they first require that the cost of each consequent be computed (which requires calls to the g-gram LM).

Restrictions to the grammar afford alternative decoding strategies to reduce the runtime cost of synchronous parsing. (Zhang et al., 2006) "binarize" grammars into CNF normal form, while (Watanabe et al., 2006) allow only Griebach-Normal form grammars. (Wellington et al., 2006) argue that these restrictions reduce our ability to model translation equivalence effectively. We take an agnostic view on the issue; directly addressing the question of efficient LM intersection rather than grammar construction.

3 Two-pass LM Intersection

We propose a two-pass solution to the problem of online g-gram LM intersection. A naive two-pass approach would simply ignore the LM interactions during parsing, extract a set of N derivations from the sentence spanning hypergraph and rescore these derivations with the g-gram LM. In practice, this approach performs poorly (Chiang, 2007; Zollmann and Venugopal, 2006). While parsing time is dramatically reduced (and N-best extraction time is negligible), N is typically significantly less than the complete number of possible derivations and substantial search errors remain. We propose an approach that builds upon the concept of a second pass but uses the g-gram LM to search for alternative, better translations.

3.1 First pass: parsing

We begin by relaxing the criterion that determines when two chart items are equivalent during parsing. We consider two chart items to be equivalent (and therefore candidates for recombination) if they have matching left-hand side nonterminals, and span. We no longer require them to have the same LM context e. We *do* however propagate the e, w for the chart item with highest score, causing the algorithm to still compute LM probabilities during parsing. As a point of notation, we refer to such a chart item by annotating its e, w as e^1, w^1, and we refer to them as approximate items (since they have made a first-best approximation for the purposes of LM calculation). These approximate items labeled with e^1, w^1 are used in consequent parse calculations.

The parsing algorithm under this approximation stays unchanged, but parsing time is dramatically reduced. The runtime complexity of this algorithm is now $\mathcal{O}\left(n^3|\mathcal{N}|^K\right)$ at the cost of significant search errors (since we ignored most LM contexts that we encountered).

This relaxation is different from approaches that do not use the LM during parsing. The sentence spanning item *does* have LM probabilities associated with it (but potentially valuable chart items were not considered during parsing). Like in traditional parsing, we retain the recombined items in the cell to allow us to explore new derivations in a second stage.

3.2 Second pass: hypergraph search

The goal item of the parsing step represents a sentence spanning hypergraph of alternative derivations. Exploring alternatives from this hypergraph and updating LM probabilities can now reveal derivations with *higher* scores that were not considered in the first pass. Exploring the whole space of alternative derivations in this hypergraph is clearly infeasible. We propose a g-gram LM driven heuristic search "H.Search" of this space that allows the g-gram LM to decide which section of the hypergraph to explore. By construction, traversing a particular derivation item from the parse chart in target-side left-to-right, depth-first order yields the correctly ordered sequence of target terminals that is the translation represented by this item. Now consider a *partial* traversal of the item in that order, where we generate only the first M target terminals, leaving the rest of the item in its original backpointer form. We informally define our second pass algorithm based on these partial derivation items.

Consider a simple example, where we have parsed a source sentence, and arrived at a sentence spanning item obtained from a rule with the following target side:

$$\mathrm{NP}_2\ \mathrm{VP}_3\ \mathrm{PP}_1$$

and that the item's best-score estimate is w. A partial traversal of this item would replace NP_2 with one of the translations available in the chart cell underlying NP_2 (called "unwinding"), and recalculate the weights associated with this item, taking into account the alternative target terminals. Assuming "the nice man" was the target side of the best scoring item in NP_2, the respective traversal would maintain the same weight. An alternative item at NP_2 might yield "a nice man". This partial traversal represents a possible item that we did not consider during parsing, and recalculating LM probabilities for this new item (based on approximate items VP_3 and PP_1) yields weight w_2:

$$\text{the nice man } \mathrm{VP}_3\ \mathrm{PP}_1 : w_1 = w$$
$$\text{a nice man } \mathrm{VP}_3\ \mathrm{PP}_1 : w_2$$

Alternative derivation items that obtain a higher score than the best-score estimates represent recovery from search errors. Our algorithm is based on the premise that these items should be traversed further, with the LM continuing to score newly generated target words. These partially traversed items are placed on an agenda (sorted by score). At each step of the second pass search, we select those items from the agenda that are within a search beam of Z from the best item, and perform the unwind operation on each of these items. Since we unwind partial items from left-to-right the g-gram LM is able to influence the search through the space of alternative derivations.

Applying the g-gram LM on partial items with leading only-terminal symbols allows the integration of high- / flexible-order LMs during this second stage process, and has the advantage of exploring only those alternatives that participate in sentence spanning, high scoring (considering both LM and translation model scores) derivations. While we do not evaluate such models here, we note that H.Search was developed specifically for the integration of such models during search.

We further note that partial items that have generated translations that differ only in the word positions up to $g-1$ words before the first nonterminal site can be recombined (for the same reasons as during LM intersected parsing). For example, when considering a 3-gram LM, the two partial items above can be recombined into one equivalence class, since partial item LM costs resulting from these items would only depend on 'nice man', but not on 'a' vs. 'the'. Even if two partial items are candidates for recombination due to their terminal words, they must also have identical backpointers (representing a set of approximate parse decisions for the rest of the sentence, in our example $\mathrm{VP}_3\mathrm{PP}_1$). Items that are filed into existing equivalence classes with a lower score are not put onto the agenda, while those that are better, or have created new equivalence classes are scheduled onto the agenda. For each newly created partial derivation, we also add a backpointer to the "parent" partial derivation that was unwound to create it.

This equivalence classing operation transforms the original left-hand side NT based hypergraph into an (ordinary) graph of partial items. Each equivalence class is a node in this new graph, and recombined items are the edges. Thus, N-best extraction can now be performed on this graph. We use the extraction method from (Huang and Chiang, 2005).

The expensive portion of our algorithm lies in the unwinding step, in which we generate a new partial item for each alternative at the non-terminal site that we are "unwinding". For each new partial item, we factor *out* LM estimates and rule weights that were used to score the parent item, and factor *in* the LM probabilities and rule weights of the alternative choice that we are considering. In addition, we must also update the new item's LM estimates for the remaining non-terminal and terminal symbols that depend on this new left context of terminals. Fortunately, the number of LM calculations per new item is constant, i.e., does not dependent on the length of the partial derivation, or how unwound it is. Only $(g-1)*2$ LM probabilities have to be re-evaluated per partial item. We now define this "unwind-recombine" algorithm formally.

3.2.1 The unwind-recombine algorithm

Going back to the first-pass parsing algorithm (Figure 1), remember that each application of a grammar rule containing nonterminals corresponds to an application of the third inference rule of the algorithm. We can assign chart items C created by the third inference rule a *back-pointer (BP) target side* as follows: When applying the third inference rule, each nonterminal occurrence $(X^k)_k$ in the corresponding $Z \rightarrow \langle \lambda, \alpha \rangle$ grammar rule corresponds to a chart cell $[X^k, i_k, j_k]$ used as an antecedent for the inference rule. We assign a BP target side for C by replacing NT occurrences in α (from the rule that created C) with backpointers to their corresponding antecedent chart cells. Further we define the distinguished backpointer P_S as the pointer to the goal cell $[S, 0, n] : w^*$.

The deductive program for our second-pass algorithm is presented in Figure 2. It makes use of two kind of items. The first, $\{P \rightarrow \alpha; e^1\} : w$, links a backpointer P to a BP target side, storing current-item vs. best-item correction terms in form of an LM context e^1 and a relative score w. The second item form $[[e; \alpha]]$ in this algorithm corresponds to partial left-to-right traversal states as described above, where e is the LM context of the traversed and unwound translation part, and α the part that is yet to be traversed and whose backpointers are still to be unwound. The first deduction rule presents the logical axioms, creating BP items for each backpointer used in a NT inference rule application during the first-pass parsing step. The second deduction rule represents the unwinding step as discussed in the example above. These deductions govern a search for derivations through the hypergraph that is driven by updates of rule weights and LM probabilities when unwinding non-first-best hypotheses. The functions p and q are as defined in Section 2, except that the domain of q is extended to BP target sides by first replacing each back-pointer with its corresponding chart cell's LM context and then applying the original q on the resulting sequence of target-terminals and $\star$ symbols.[2] Note that w', which was computed by the first deduction rule, adjusts the current hypothesis' weight

[2]Note also that $p(\langle s \rangle^{g-1} \star \langle \backslash s \rangle^{g-1}) = 1$ as the product over the empty set is one.

$$\frac{}{\{P \rightarrow \alpha; e^1\} : w'/w} \quad (P \text{ back-points to 1st-pass cell } [X, i, j; e^1] : w;\ \alpha \text{ and } w' \text{ are BP-target-side and weight of one of that cell's items})$$

$$\frac{[[e; P\alpha_{end}]] : w \quad \{P \rightarrow \alpha_{lex}\alpha_{mid}; e^1\} : w'}{[[q(e\alpha_{lex}); \alpha_{mid}\alpha_{end}]] : ww'p[eq(\alpha_{lex}\alpha_{mid})]p[q(e\alpha_{lex}\alpha_{mid})q(\alpha_{end})]/p(ee^1)/p[q(ee^1)q(\alpha_{end})]} \left(\begin{array}{c} \alpha_{lex} \text{ contains no BPs and} \\ \alpha_{mid} = P'\alpha' \text{ or } \alpha_{mid} = \varepsilon \end{array} \right)$$

Figure 2. Left-to-right LM driven hypergraph search of the sentence spanning hypergraph; ε denotes the empty word. Non-logical (*Start*) axiom: $[[\varepsilon; P_S]] : w^*$; Goal item: $[[\langle s\rangle^{g-1} \star \langle \backslash s\rangle^{g-1}; \varepsilon]] : w$

that is based on the first-best instance of P to the actually chosen instance's weight. Further, the ratio $p(eq(\alpha_{lex}\alpha_{mid}))/p(ee^1)$ adjusts the LM probabilities of P's instantiation given its left context, and $p[q(e\alpha_{lex}\alpha_{mid})q(\alpha_{end})]/p[q(ee^1)q(\alpha_{end})]$ adjusts the LM probabilities of the $g-1$ words right of P.

4 Alternative Approaches

We evaluate our two pass hypergraph search "H.Search" against the strong single pass Cube Pruning (CP) baseline as mentioned in (Chiang, 2005) and detailed in (Chiang, 2007). In the latter work, the author shows that CP clearly outperforms both the naive single pass solution of severe pruning as well as the naive two-pass rescoring approach. Thus, we focus on comparing our approach to CP.

CP is an optimization to the intersected LM parsing algorithm presented in Figure 1. It addresses the creation of the $\prod_{k=1}^{K} \mid [X_k, i_k, j_k, *] \mid$ chart items when generating consequent items. CP amounts to an early termination condition when generating the set of possible consequents. Instead of generating all consequents, and then pruning away the poor performers, CP uses the K-Best extraction approach of (Huang and Chiang, 2005) to select the best K consequents only, at the cost of potential search errors. CP's termination condition can be defined in terms of an absolute number of consequents to generate, or by terminating the generation process when a newly generated item is worse (by β) than the current best item for the same left-hand side and span. To simulate comparable pruning criteria we parameterize each method with soft-threshold based criteria only (β for CP and Z for H.Search) since counter based limits like K have different effects in CP (selecting e labeled items) vs H.Search (selecting rules since items are not labeled with e).

5 Experimental Framework

We present results on the IWSLT 2006 Chinese to English translation task, based on the Full BTEC corpus of travel expressions with 120K parallel sentences (906K source words and 1.2m target words). The evaluation test set contains 500 sentences with an average length of 10.3 source words.

Grammar rules were induced with the syntax-based SMT system "SAMT" described in (Zollmann and Venugopal, 2006), which requires initial phrase alignments that we generated with "GIZA++" (Koehn et al., 2003), and syntactic parse trees of the target training sentences, generated by the Stanford Parser (D. Klein, 2003) pre-trained on the Penn Treebank. All these systems are freely available on the web.

We experiment with 2 grammars, one syntax-based (3688 nonterminals, 0.3m rules), and one purely hierarchical (1 generic nonterminal, 0.05m rules) as in (Chiang, 2005). The large number of nonterminals in the syntax based systems is due to the CCG extension over the original 75 Penn Treebank nonterminals. Parameters λ used to calculate $P(D)$ are trained using MER training (Och, 2003) on development data.

6 Comparison of Approaches

We evaluate each approach by considering both search errors made on the development data for a fixed set of model parameters, and the BLEU metric to judge translation quality.

6.1 Search Error Analysis

While it is common to evaluate MT quality using the BLEU score, we would like to evaluate search errors made as a function of "effort" made by each algorithm to produce a first-best translation. We consider two metrics of effort made by each algorithm.

We first evaluate search errors as a function of novel queries made to the g-gram LM (since LM calls tend to be the dominant component of runtime in large MT systems). We consider novel queries as those that have not already been queried for a particular sentence, since the repeated calls are typically efficiently cached in memory and do not affect runtime significantly. Our goal is to develop techniques that can achieve low search error with the fewest novel queries to the g-gram LM.

To appreciate the practical impact of each algorithm, we also measure search errors as a function of the number of seconds required to translate a fixed unseen test set. This second metric is more sensitive to implementation and, as it turned out, even compiler memory management decisions.

We define search errors based on the weight of the best sentence spanning item. Treating weights as negative log probabilities (costs), we accumulate the value of the lowest cost derivation for each sentence in the testing data as we vary pruning settings appropriate to each method. Search errors are reduced when we are able to lower this accumulated model cost. We prefer approaches that yield low model cost with the least number of LM calls or number of seconds spent decoding.

It is important to note that model cost $(-\log(P(D)))$ is a function of the parameters λ which have been trained using MER training. The parameters used for these experiments were trained with the CP approach; in practice we find that either approach is effective for MER training.

6.2 Results

Figure 3 and Figure 4 plot model cost as a function of LM cache misses for the IWSLT Hierarchical and Syntax based systems, while Figure 5 plots decoding time. The plots are based on accumulated model cost, decoding time and LM cache misses over the IWSLT Test 06 set. For H.Search, we vary the beam parameter Z for a fixed value of $\beta = 5$ during parsing while for CP, we vary β. We also limit the total number of items on the agenda at any time to 1000 for H.Search as a memory consideration. We plot each method until we see no change in BLEU score for that method. BLEU scores for each parameter setting are also noted on the plots.

For both the hierarchical and syntax based grammars we see that the H.Search method achieves a given model cost 'earlier' in terms of novel LM calls for most of the plotted region, but ultimately fails to achieve the same lowest model cost as the CP method.[3] While the search beam of Z mitigates the impact of the estimated scores during H.Search's second pass, the score is still not an admissible heuristic for error-free search. We suspect that simple methods to "underestimate" the score of a partial derivation's remaining nonterminals could bridge this gap in search error. BLEU scores in the regions of lowest model cost tend to be reasonably stable and reflect comparable translation performance for both methods.

Under both H.Search and CP, the hierarchical grammar ultimately achieves a BLEU score of 19.1%, while the syntactic grammar's score is approximately 1.5 points higher at 20.7%. The hierarchical grammar demonstrates a greater variance of BLEU score for both CP and H.Search compared to the syntax-based grammar. The use of syntactic structure serves as an additional model of target language fluency, and can explain the fact that syntax based translation quality is more robust to differences in the number of g-gram LM options explored.

Decoding time plots shows a similar result, but with diminished relative improvement for the H.Search method. Profiling analysis of the H.Search method shows that significant time is spent simply on allocating and deallocating memory for partial derivations on top of the scoring times for these items. We expect to be able to reduce this overhead significantly in future work.

7 Conclusion

We presented an novel two-pass decoding approach for PSCFG-based machine translation that achieves search errors comparable to the state of the art Cube Pruning method. By maintaining comparable, sentence spanning derivations we allow easy integration of high or flexible order LMs as well as sentence level syntactic features during the search process. We plan to evaluate the impact of these more powerful models in future work. We also hope to address the question of how much search error is tolerable to

[3]Analysis of *total* LM calls made by each method (not presented here) shows the H.Search makes significantly fewer (1/2) total LM calls than CP to achieve each model cost.

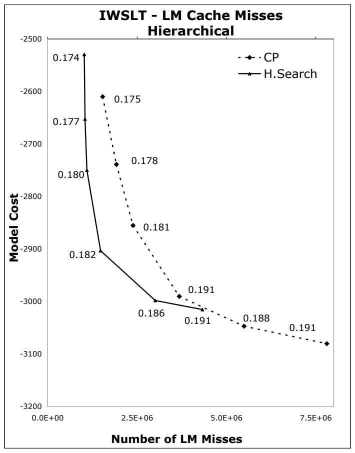


Figure 3. LM caches misses for IWSLT hierarchical grammar and BLEU scores for varied pruning parameters

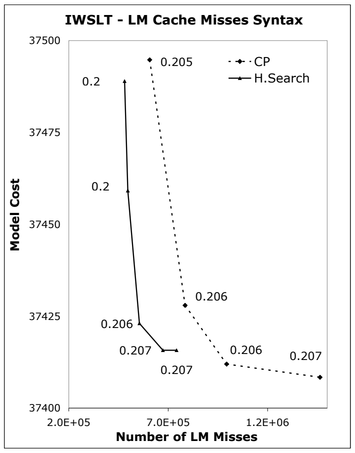


Figure 4. LM caches misses for IWSLT syntactic grammar and BLEU scores for varied pruning parameters

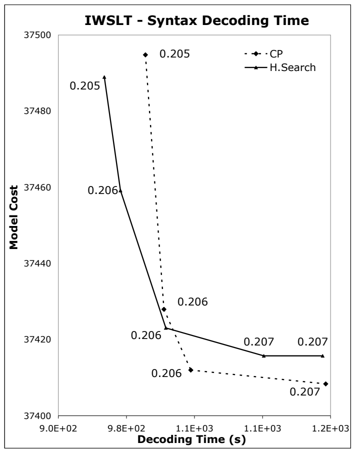


Figure 5. Decoding time (s) for IWSLT syntactic grammar and BLEU scores for varied pruning parameters

run MER training and still generate parameters that generalize well to test data. This point is particularly relevant to evaluate the use of search error analysis.

References

Bar-Hillel, M.Perles, and E.Shamir. 1964. An efficient context-free parsing algorithm. *Communications of the Assocation for Computing Machinery*.

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

David Chiang. 2007. Hierarchical phrase based translation. *To Appear in the Journal of Computational Linguistics*.

C. Manning D. Klein. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

Matthias Eck and Chiori Hori. 2005. Overview of the IWSLT 2005 evaluation campaign. In *Proc. of International Workshop on Spoken Language Translation*, pages 11–17.

Michael Galley, M. Hopkins, Kevin Knight, and Daniel Marcu. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proc. of NAACL-HLT*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. of the 9th International Workshop on Parsing Technologies*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT/NAACL*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, Sapporo, Japan, July 6-7.

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

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–15.

Mark Steedman. 1999. Alternative quantifier scope in CCG. In *Proc. of ACL*.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase based translation. In *ACL*.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *ACL*.

Dekai Wu. 1996. A polynomial time algorithm for statistical machine translation. In *Proc. of the Association for Computational Linguistics*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. of HLT/NAACL*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of the Workshop on Statistical Machine Translation, HLT/NAACL*, New York, June.

Statistical Phrase-based Post-editing

Michel Simard **Cyril Goutte** **Pierre Isabelle**
Interactive Language Technologies
National Research Council of Canada
Gatineau, Canada, K1A 0R6
FirstName.LastName@nrc.gc.ca

Abstract

We propose to use a statistical phrase-based machine translation system in a *post-editing* task: the system takes as input raw machine translation output (from a commercial rule-based MT system), and produces post-edited target-language text. We report on experiments that were performed on data collected in precisely such a setting: pairs of raw MT output and their manually post-edited versions. In our evaluation, the output of our *automatic post-editing* (APE) system is not only better quality than the rule-based MT (both in terms of the BLEU and TER metrics), it is also better than the output of a state-of-the-art phrase-based MT system used in standalone translation mode. These results indicate that automatic post-editing constitutes a simple and efficient way of combining rule-based and statistical MT technologies.

1 Introduction

The quality of machine translation (MT) is generally considered insufficient for use in the field without a significant amount of human correction. In the translation world, the term *post-editing* is often used to refer to the process of manually correcting MT output. While the conventional wisdom is that post-editing MT is usually not cost-efficient compared to full human translation, there appear to be situations where it is appropriate and even profitable. Unfortunately, there are few reports in the literature about such experiences (but see Allen (2004) for examples).

One of the characteristics of the post-editing task, as opposed to the revision of human translation for example, is its partly repetitive nature. Most MT systems invariably produce the same output when confronted with the same input; in particular, this means that they tend to make the same mistakes over and over again, which the post-editors must correct repeatedly. Batch corrections are sometimes possible when multiple occurrences of the same mistake appear in the same document, but when it is repeated over several documents, or equivalently, when the output of the same machine translation system is handled by multiple post-editors, then the opportunities for factoring corrections become much more complex. MT users typically try to reduce the post-editing load by customizing their MT systems. However, in Rule-based Machine Translation (RBMT), which still constitutes the bulk of the current commercial offering, customization is usually restricted to the development of "user dictionaries". Not only is this time-consuming and expensive, it can only fix a subset of the MT system's problems.

The advent of Statistical Machine Translation, and most recently phrase-based approaches (PBMT, see Marcu and Wong (2002), Koehn et al. (2003)) into the commercial arena seems to hold the promise of a solution to this problem: because the MT system learns directly from existing translations, it can be automatically customized to new domains and tasks. However, the success of this operation crucially depends on the amount of training data available. Moreover, the current state of the technology is still insufficient for consistently producing human readable translations.

This state of affairs has prompted some to examine the possibility of automating the post-editing process itself, at least as far as "repetitive errors" are concerned. Allen and Hogan (2000) sketch the outline of such an automated post-editing (APE) system, which would automatically learn post-editing rules from a *tri-parallel* corpus of source, raw MT and post-edited text. Elming (2006) suggests using tranformation-based learning to automatically acquire error-correcting rules from such data; however, the proposed method only applies to lexical choice errors. Knight and Chander (1994) also argue in favor of using a separate APE module, which is then portable across multiple MT systems and language pairs, and suggest that the post-editing task could be performed using statistical machine translation techniques. To the best of our knowledge, however, this idea has never been implemented.

In this paper, we explore the idea of using a PBMT system as an automated post-editor. The underlying intuition is simple: if we collect a parallel corpus of raw machine-translation output, along with its human-post-edited counterpart, we can train the system to translate from the former into the latter. In section 2, we present the case study that motivates our work and the associated data. In section 3, we describe the phrase-based post-editing model that we use for improving the output of the automatic translation system. In section 4, we illustrate this on a dataset of moderate size containing job ads and their translation. With less than 500k words of training material, the phrase-based MT system already outperforms the rule-based MT baseline. However, a phrase-based post-editing model trained on the output of that baseline outperforms both by a fairly consistent margin. The resulting BLEU score increases by up to 50% (relative) and the TER is cut by one third.

2 Background

2.1 Context

The Canadian government's department of Human Resources and Social Development (HRSDC) maintains a web site called *Job Bank*,[1] where potential employers can post ads for open positions in Canada. Over one million ads are posted on *Job Bank* every year, totalling more than 180 million words. By virtue of Canada's Official Language Act, HRSDC is under legal obligation to post all ads in both French and English. In practice, this means that ads submitted in English must be translated into French, and vice-versa.

To address this task, the department has put together a complex setup, involving text databases, translation memories, machine translation and human post-editing. Employers submit ads to the *Job Bank* website by means of HTML forms containing "free text" data fields. Some employers do periodical postings of identical ads; the department therefore maintains a database of previously posted ads, along with their translations, and new ads are systematically checked against this database. The translation of one third of all ads posted on the *Job Bank* is actually recuperated this way. Also, employers will often post ads which, while not entirely identical, still contain identical sentences. The department therefore also maintains a translation memory of individual sentence pairs from previously posted ads; another third of all text is typically found *verbatim* in this way.

The remaining text is submitted to machine translation, and the output is post-edited by human experts. Overall, only a third of all submitted text requires human intervention. This is nevertheless very labour-intensive, as the department tries to ensure that ads are posted at most 24 hours after submission. The *Job Bank* currently employs as many as 20 post-editors working full-time, most of whom are junior translators.

2.2 The Data

HRSDC kindly provided us with a sample of data from the *Job Bank*. This corpus consists in a collection of parallel "blocks" of textual data. Each block contains three parts: the source language text, as submitted by the employer, its machine-translation, produced by a commercial rule-based MT system, and its final post-edited version, as posted on the website.

[1] `http://www.jobbank.gc.ca`

The entire corpus contains less than one million words in each language. This corresponds to the data processed in less than a week by the *Job Bank*. Basic statistics are given in Table 1 (see Section 4.1). Most blocks contain only one sentence, but some blocks may contain many sentences. The longest block contains 401 tokens over several sentences. Overall, blocks are quite short: the median number of tokens per source block is only 9 for French-to-English and 7 for English-to-French. As a consequence, no effort was made to segment the blocks further for processing.

We evaluated the quality of the Machine Translation contained in the corpus using the Translation Edit Rate (TER, cf. Snover et al. (2006)). The TER counts the number of edit operations, including phrasal shifts, needed to change a hypothesis translation into an adequate and fluent sentence, and normalised by the length of the final sentence. Note that this closely corresponds to the post-editing operation performed on the *Job Bank* application. This motivates the choice of TER as the main metric in our case, although we also report BLEU scores in our experiments. Note that the emphasis of our work is on reducing the *post-edition effort*, which is well estimated by TER. It is not directly on *quality* so the question of which metric better estimates translation quality is not so relevant here.

The global TER (over all blocks) are 58.77% for French-to-English and 53.33% for English-to-French. This means that more than half the words have to be post-edited in some way (delete / substitute / insert / shift). This apparently harsh result is somewhat mitigated by two factors.

First, the distribution of the block-based TER[2] shows a large disparity in performance, cf. Figure 1. About 12% of blocks have a TER higher than 100%: this is because the TER normalises on the length of the references, and if the raw MT output is longer than its post-edited counterpart, then the number of edit operations may be larger than that length.[3] At the other end of the spectrum, it is also clear that many blocks have low TER. In fact more than 10% have a TER of 0. The global score therefore hides a large range of performance.

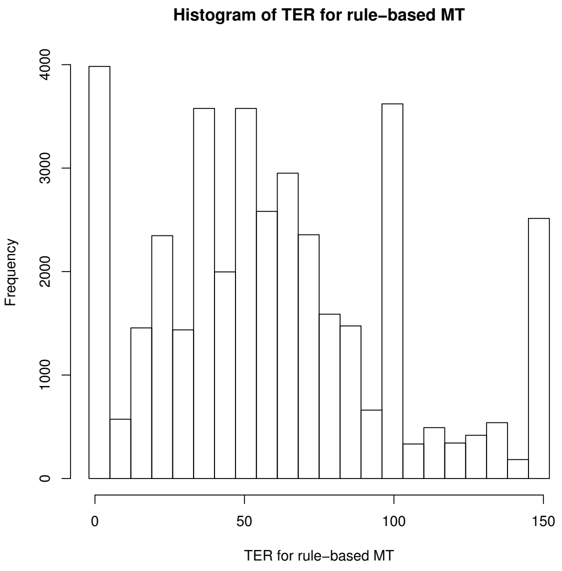


Figure 1: Distribution of TER on 39005 blocks from the French-English corpus (thresholded at 150%).

The second factor is that the TER measures the distance to an adequate *and fluent* result. A high TER does not mean that the raw MT output is not understandable. However, many edit operations may be needed to make it fluent.

3 Phrase-based Post-editing

Translation post-editing can be viewed as a simple transformation process, which takes as input raw target-language text coming from a MT system, and produces as output target-language text in which "errors" have been corrected. While the automation of this process can be envisaged in many different ways, the task is not conceptually very different from the translation task itself. Therefore, there doesn't seem to be any good reason why a machine translation system could not handle the post-editing task. In particular, given such data as described in Section 2.2, the idea of using a statistical MT system for post-editing is appealing. *Portage* is precisely such a system, which we describe here.

Portage is a phrase-based, statistical machine translation system, developed at the National Research Council of Canada (NRC) (Sadat et al.,

[2]Contrary to BLEU or NIST, the TER naturally decomposes into block-based scores.

[3]A side effect of the normalisation is that larger TER are measured on small sentences, e.g. 3 errors for 2 reference words.

2005). A version of the Portage system is made available by the NRC to Canadian universities for research and education purposes. Like other SMT systems, it learns to translate from existing parallel corpora.

The system translates text in three main phases: preprocessing of raw data into tokens; decoding to produce one or more translation hypotheses; and error-driven rescoring to choose the best final hypothesis. For languages such as French and English, the first of these phases (tokenization) is mostly a straightforward process; we do not describe it any further here.

Decoding is the central phase in SMT, involving a search for the hypotheses t that have highest probabilities of being translations of the current source sentence s according to a model for $P(t|s)$. Portage implements a dynamic programming beam search decoding algorithm similar to that of Koehn (2004), in which translation hypotheses are constructed by combining in various ways the target-language part of phrase pairs whose source-language part matches the input. These phrase pairs come from large *phrase tables* constructed by collecting matching pairs of contiguous text segments from word-aligned bilingual corpora.

Portage's model for $P(t|s)$ is a log-linear combination of four main components: one or more n-gram target-language models, one or more phrase translation models, a distortion (word-reordering) model, and a sentence-length feature. The phrase-based translation model is similar to that of Koehn, with the exception that phrase probability estimates $P(\tilde{s}|\tilde{t})$ are smoothed using the Good-Turing technique (Foster et al., 2006). The distortion model is also very similar to Koehn's, with the exception of a final cost to account for sentence endings.

Feature function weights in the loglinear model are set using Och's minium error rate algorithm (Och, 2003). This is essentially an iterative two-step process: for a given set of source sentences, generate n-best translation hypotheses, that are representative of the entire decoding search space; then, apply a variant of Powell's algorithm to find weights that optimize the BLEU score over these hypotheses, compared to reference translations. This process is repeated until the set of translations stabilizes, i.e. no new translations are produced at the decoding step.

To improve raw output from decoding, Portage relies on a rescoring strategy: given a list of n-best translations from the decoder, the system reorders this list, this time using a more elaborate loglinear model, incorporating more feature functions, in addition to those of the decoding model: these typically include IBM-1 and IBM-2 model probabilities (Brown et al., 1993) and an IBM-1-based feature function designed to detect whether any word in one language appears to have been left without satisfactory translation in the other language; all of these feature functions can be used in both language directions, i.e. source-to-target and target-to-source.

In the experiments reported in the next section, the Portage system is used both as a translation and as an APE system. While we can think of a number of modifications to such a system to better adapt it to the post-editing task (some of which are discussed later on), we have done no such modifications to the system. In fact, whether the system is used for translation or post-editing, we have used exactly the same translation model configuration and training procedure.

4 Evaluation

4.1 Data and experimental setting

The corpus described in section 2.2 is available for two language pairs: English-to-French and French-to-English.[4] In each direction, each block is available in three versions (or *slices*): the original text (or *source*), the output of the commercial rule-based MT system (or *baseline*) and the final, post-edited version (or *reference*).

In each direction (French-to-English and English-to-French), we held out two subsets of approximately 1000 randomly picked blocks. The *validation* set is used for testing the impact of various high-level choices such as pre-processing, or for obtaining preliminary results based on which we setup new experiments. The *test* set is used only once, in order to obtain the final experimental results reported here.

The rest of the data constitutes the *training* set, which is split in two. We sampled a subset of 1000 blocks as *train-2*, which is used for optimiz-

[4]Note that, in a post-editing context, translation direction is crucially important. It is not possible to use the same corpus in both directions.

Corpus	English-to-French				French-to-English			
		words:				words:		
	blocks	source	baseline	reference	blocks	source	baseline	reference
train-1	28577	310k	382k	410k	36005	485k	501k	456k
train-2	1000	11k	14k	14k	1000	13k	14k	12k
validation	881	10k	13k	13k	966	13k	14k	12k
test	899	10k	12k	13k	953	13k	13k	12k

Table 1: Data and split used in our experiments, (in thousand words). 'baseline' is the output of the commercial rule-based MT system and 'reference' is the final, post-edited text.

ing the log-linear model parameters used for decoding and rescoring. The rest is the *train-1* set, used for estimating IBM translation models, constructing phrasetables and estimating a target language model.

The composition of the various sets is detailed in Table 1. All data was tokenized and lowercased; all evaluations were performed independent of case. Note that the *validation* and *test* sets were originally made out of 1000 blocks sampled randomly from the data. These sets turned out to contain blocks identical to blocks from the training sets. Considering that these would normally have been handled by the translation memory component (see the HRSDC workflow description in Section 2.1), we removed those blocks for which the source part was already found in the *training* set (in either *train-1* or *train-2*), hence their smaller sizes.

In order to check the sensitivity of experimental results to the choice of the *train-2* set, we did a run of preliminary experiments using different subsets of 1000 blocks. The experimental results were nearly identical and highly consistent, showing that the choice of a particular *train-2* subset has no influence on our conclusions. In the experiments reported below, we therefore use a single identical *train-2* set.

We initially performed two sets of experiments on this data. The first was intended to compare the performance of the Portage PBMT system as an alternative to the commercial rule-based MT system on this type of data. In these experiments, English-to-French and French-to-English translation systems were trained on the source and reference (manually post-edited target language) slices of the training set. In addition to the target language model estimated on the *train-1* data, we used an external contribution,

Language	TER	BLEU
English-to-French		
Baseline	53.5	32.9
Portage *translation*	53.7	36.0
Baseline + Portage APE	**47.3**	**41.6**
French-to-English		
Baseline	59.3	31.2
Portage *translation*	43.9	41.0
Baseline + Portage APE	**41.0**	**44.9**

Table 2: Experimental Results: For TER, lower (error) is better, while for BLEU, higher (score) is better. Best results are in bold.

a trigram target language model trained on a fairly large quantity of data from the Canadian Hansard.

The goal of the second set of experiments was to assess the potential of the Portage technology in automatic post-editing mode. Again, we built systems for both language directions, but this time using the existing rule-based MT output as source and the reference as target. Apart from the use of different source data, the training procedure and system configurations of the translation and post-editing systems were in all points identical.

4.2 Experimental results

The results of both experiments are presented in Table 2. Results are reported both in terms of the TER and BLEU metrics; *Baseline* refers to the commercial rule-based MT output.

The first observation from these results is that, while the performance of Portage in translation mode is approximately equivalent to that of the baseline system when translating into French, its performance is much better than the baseline when translating into English. Two factors possibly contribute to this result: first, the fact that the baseline system itself performs better when translating into French; second, and possibly more importantly, the fact that we had access to less training data for English-to-French translation.

The second observation is that when Portage is used in automatic post-editing mode, on top of the baseline MT system, it achieves better quality than either of the two translation systems used on its own. This appears to be true regardless of the translation direction or metric. This is an extremely interesting result, especially in light of how little data was actually available to train the post-editing system.

One aspect of statistical MT systems is that, contrary to rule-based systems, their performance (usually) increases as more training data is available. In order to quantify this effect in our setting, we have computed learning curves by training the Portage translation and Portage APE systems on subsets of the training data of increasing sizes. We start with as little as 1000 blocks, which corresponds to around 10-15k words.

Figure 2 (next page) compares the learning rates of the two competing approaches (Portage translation vs. Portage APE). Both approaches display very steady learning rates (note the logarithmic scale for training data size). These graphs strongly suggest that both systems would continue to improve given more training data. The most impressive aspect is how little data is necessary to improve upon the baseline, especially when translating into English: as little as 8000 blocks (around 100k words) for direct translation and 2000 blocks (around 25k words) for automatic post-editing. This suggests that such a post-editing setup might be worth implementing even for specialized domains with very small volumes of data.

4.3 Extensions

Given the encouraging results of the Portage APE approach in the above experiments, we were curious to see whether a Portage+Portage combination might be as successful: after all, if Portage was good at correcting some other system's output, could it not manage to correct the output of another Portage translator?

We tested this in two settings. First, we actually use the output of the Portage translation system obtained above, i.e. trained on the same data. In our second experiment, we use the output of a Portage translator trained on different domain data (the Canadian Hansard), but with much larger amounts of training material (over 85 million words per language). In both sets of experiments, the Portage APE system was trained as previously, but using Portage translations of the Job Bank data as input text.

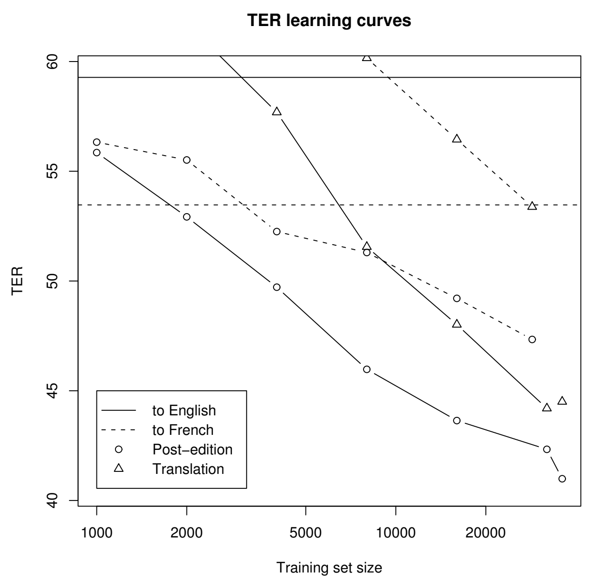
Language	TER	BLEU
English-to-French		
Portage *Job Bank*	**53.7**	36.0
+ Portage APE	**53.7**	**36.2**
Portage *Hansard*	76.9	13.0
+ Portage APE	64.6	26.2
French-to-English		
Portage *Job Bank*	**43.9**	41.0
+ Portage APE	**43.9**	**41.4**
Portage *Hansard*	80.1	14.0
+ Portage APE	57.7	28.6

Table 3: Portage translation - Portage APE system combination experimental results.

The results of both experiments are presented in Table 3. The first observation in these results is that there is nothing to be gained from post-editing when both the translation and APE systems are trained on the same data sets (Portage *Job Bank* + Portage APE experiments). In other words, the translation system is apparently already making the best possible use of the training data, and additional layers do not help (but nor do they hurt, interestingly).

However, when the translation system has been trained using distinct data (Portage *Hansard* + Portage APE experiments), post-editing makes a large difference, comparable to that observed with the rule-based MT output provided with the *Job Bank* data. In this case, however, the Portage translation system behaves very poorly in spite of the important size of the training set for this system, much worse in fact than the "baseline" system. This highlights the fact that both the *Job Bank* and *Hansard* data are very much domain-specific, and that access to appropriate training material is crucial for phrase-based translation technology.

In this context, combining two phrase-based sys-



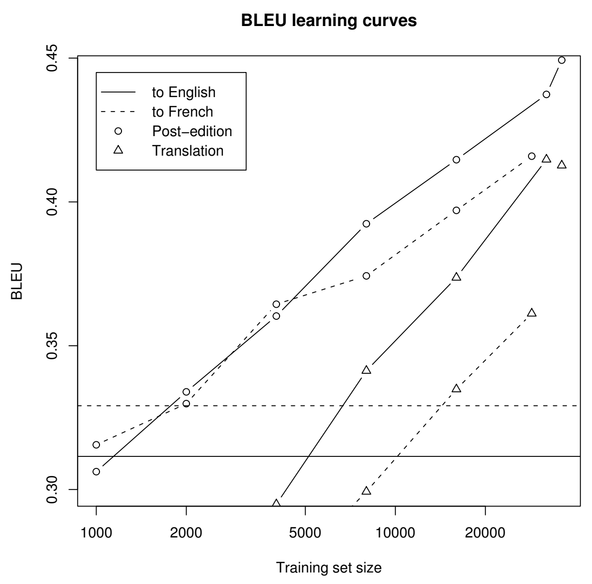


Figure 2: TER and BLEU scores of the phrase-based post-editing models as the amount of training data increases (log scale). The horizontal lines correspond to the performance of the baseline system (rule-based translation).

tems as done here can be seen as a way of adapting an existing MT system to a new text domain; the APE system then acts as an "adapter", so to speak. Note however that, in our experiments, this setup doesn't perform as well as a single Portage translation system, trained directly and exclusively on the *Job Bank* data.

Such an adaptation strategy should be contrasted with one in which the translation models of the old and new domains are "merged" to create a new translation system. As mentioned earlier, Portage allows using multiple phrase translation tables and language models concurrently. For example, in the current context, we can extract phrase tables and language models from the *Job Bank* data, as when training the "Portage *Job Bank*" translation system, and then build a Portage translation model using both the *Hansard* and *Job Bank* model components. Loglinear model parameters are then optimized on the *Job Bank* data, so as to find the model weights that best fit the new domain.

In a straightforward implementation of this idea, we obtained performances almost identical to those of the Portage translation system trained solely on *Job Bank* data. Upon closer examination of the model parameters, we observed that *Hansard* model components (language model, phrase tables, IBM translation models) were systematically attributed negligeable weights. Again, the amount of training material for the new domain may be critical in chosing between alternative adaptation mechanisms.

5 Conclusions and Future Work

We have proposed using a phrase-based MT system to automatically post-edit the output of another MT system, and have tested this idea with the Portage MT system on the *Job Bank* data set, a corpus of manually post-edited French-English machine translations. In our experiments, not only does phrase-based APE significantly improve the quality of the output translations, this approach outperforms a standalone phrase-based translation system.

While these results are very encouraging, the learning curves of Figure 2 suggest that the output quality of the PBMT systems increases faster than that of the APE systems as more data is used for training. So while the combination strategy clearly performs better with limited amounts of training data, there is reason to believe that, given sufficient training data, it would eventually be outperformed by a direct phrase-based translation strategy. Of course, this remains to be verified empirically, something which will obviously require more data than is currently available to us. But this sort of behavior is expectable: while both types of system improve as more training data is used, inevitably some details of the source text will be lost by the front-end MT system, which the APE system will never be able to retrieve.[5] Ultimately, the APE system will be weighted down by the inherent limitations of the front-end MT system.

One way around this problem would be to modify the APE system so that it not only uses the baseline MT output, but also the source-language input. In the Portage system, this could be achieved, for example, by introducing feature functions into the log-linear model that relate target-language phrases with the source-language text. This is one research avenue that we are currently exploring.

Alternatively, we could combine these two inputs differently within Portage: for example, use the source-language text as the *primary* input, and use the raw MT output as a secondary source. In this perspective, if we have multiple MT systems available, nothing precludes using *all* of them as providers of secondary inputs. In such a setting, the phrase-based system becomes a sort of *combination MT system*. We intend to explore such alternatives in the near future as well.

Acknowledgements

The work reported here was part of a collaboration between the National Research Council of Canada and the department of Human Resources and Social Development Canada. Special thanks go to Souad Benayyoub, Jean-Frédéric Hübsch and the rest of the *Job Bank* team at HRSDC for preparing data that was essential to this project.

References

Jeffrey Allen and Christofer Hogan. 2000. Toward the development of a post-editing module for Machine Translation raw output: a new productivity tool for processing controlled language. In *Third International Controlled Language Applications Workshop (CLAW2000)*, Washington, USA.

Jeffrey Allen. 2004. Case study: Implementing MT for the translation of pre-sales marketing and post-sales software deployment documentation. In *Proceedings of AMTA-2004*, pages 1–6, Washington, USA.

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.

Jakob Elming. 2006. Transformation-based corrections of rule-based MT. In *Proceedings of the EAMT 11th Annual Conference*, Oslo, Norway.

George Foster, Roland Kuhn, and Howard Johnson. 2006. Phrasetable Smoothing for Statistical Machine Translation. In *Proceedings of EMNLP 2006*, pages 53–61, Sydney, Australia.

Kevin Knight and Ishwar Chander. 1994. Automated Postediting of Documents. In *Proceedings of National Conference on Artificial Intelligence*, pages 779–784, Seattle, USA.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, Edmonton, Canada.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of AMTA 2004*, pages 115–124, Washington, USA.

Daniel Marcu and William Wong. 2002. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of EMNLP 2002*, Philadelphia, USA.

Franz Josef Och. 2003. Minimum error rate training in Statistical Machine Translation. In *Proceedings of ACL-2003*, pages 160–167, Sapporo, Japan.

Fatiha Sadat, Howard Johnson, Akakpo Agbago, George Foster, Roland Kuhn, Joel Martin, and Aaron Tikuisis. 2005. PORTAGE: A Phrase-Based Machine Translation System. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 129–132, Ann Arbor, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA-2006*, Cambridge, USA.

[5]As a trivial example, imagine an MT system that "deletes" out-of-vocabulary words.

Automatic Answer Typing for How-Questions

Christopher Pinchak and **Shane Bergsma**
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2E8, Canada
{pinchak,bergsma}@cs.ualberta.ca

Abstract

We introduce an answer typing strategy specific to quantifiable *how* questions. Using the web as a data source, we automatically collect answer units appropriate to a given how-question type. Experimental results show answer typing with these units outperforms traditional fixed-category answer typing and other strategies based on the occurrences of numerical entities in text.

1 Introduction

Question answering (QA) systems are emerging as a viable means of obtaining specific information in the face of large availability. Answer typing is an important part of QA because it allows the system to greatly reduce the number of potential answers, using general knowledge of the answer form for a specific question. For example, for *what*, *where*, and *who* questions like "What is the capital of Canada?", answer typing can filter the phrases which might be proposed as candidate answers, perhaps only identifying those textual entities known to be cities.

We focus on answer typing for *how*-questions, a subset of questions which have received little specific attention in the QA community. Rather than seeking an open-ended noun or verb phrase, how-questions often seek a numerical measurement expressed in terms of a certain kind of unit, as in the following example:

Example 1: "How heavy is a grizzly bear?"

An answer typing system might expect answers to include units like *kilograms*, *pounds*, or *tons*. Entities with inappropriate units, such as *feet*, *meters*, or *honey pots*, would be excluded as candidate answers.

We specifically handle the subset of how-questions that we call *how-adjective* questions; that is, questions of the form "How *adjective*...?" such as Example 1. In particular, we do not address "how many" questions, which usually specify the units directly following *many*, nor "how much" questions, which generally seek a monetary value.

Hand-crafting a comprehensive list of units appropriate to many different adjectives is time-consuming and likely to miss important units. For example, an annotator might miss *gigabytes* for a measure of "how large." Instead of compiling a list manually, we propose a means of automatically generating lists of appropriate units for a number of real-world questions.

How-adjective questions represent a significant portion of queries sent to search engines; of the 35 million queries in the AOL search query data set (Pass et al., 2006), over 11,000 are of the form "how *adjective*..." – close to one in every three thousand queries. Of those 11,000 queries, 152 different adjectives are used, ranging from the expected "how old" and "how far" to the obscure "how orwellian."

This high proportion of queries is especially striking given that search engines provide little support for answering how-adjective questions. Indeed, most IR systems work by keyword matching. Entering Example 1 into a search engine returns documents discussing the grizzly's "heavy fur," "heavy, shaggy coat" and "heavy stout body." When faced with such results, a smart search engine user knows to inject answer units into their query to refine their search, perhaps querying "grizzly pounds." They may also convert their adjective (*heavy*) to a related concept (*weight*), for the query "grizzly weight."

Similarly, our approach discovers unit types by first converting the adjective to a related concept, using information in a structured ontology. For example, "big" can be used to obtain "size," and "tall" can derive "height." We then use an online search engine to automatically find units appropriate to the concept, given the assumption that the concept is explicitly measured in terms of specific units, e.g., height can be measured in feet, weight can be measured in pounds, and size can be measured in gigabytes.

By automatically extracting units, we do not require a set of prior questions with associated answers. Instead, we use actual questions as a source of realistic adjectives only. This is important because while large sets of existing questions can be obtained (Li and Roth, 2002), there are many fewer questions with available answers.

Our experiments demonstrate that how-question-specific unit lists consistently achieve higher answer identification performance than fixed-type, general-purpose answer typing (which propose all numerical entities as answer candidates). Furthermore, our precomputed, automatically-generated unit lists are shown to consistently achieve better performance than baseline systems which derive unit lists at runtime from documents relevant to the answer query, even when such documents are gathered using perfect knowledge of the answer distribution.

The outline of the paper is as follows. In Section 2 we outline related work. In Section 3 we provide the framework of our answer-typing model. Section 4 describes the implementation details of the model. Section 5 describes our experimental methodology, while Section 6 shows the benefits of using automatic how-question answer-typing. We conclude with possible directions of future research opened by this novel problem formulation.

2 Previous Work

Answer typing is an important component of any QA system, but varies greatly in the approach taken (Prager et al., 2003; Harabagiu et al., 2005). Basically, answer typing provides a means of filtering answer candidates as either appropriate or inappropriate to the question. For example, Li and Roth (2002) assign one of fifty possible types to a question based on features present in the question. Answer candidates can then be selected from text by finding entities whose type matches that of the input question. Similarly, Ittycheriah et al. (2000) assign one of the MUC named-entity types to each input question. In these fixed-category approaches, how-questions are assigned a fixed type in the same manner as other questions. For how-questions, this corresponds to a numerical type. However, retrieving all numerical entities will provide lower answer identification precision than a system that only provides those specified with the expected answer units.

Pinchak and Lin (2006) propose a dynamic answer typing system which computes a unique score for the appropriateness of any word to a particular question. Unfortunately, their question context-mapping is limited to *what*, *where*, and *who* questions, and thus is not defined for how-questions.

Wu et al. (2005) handle how-questions differently than other questions. They use special hand-crafted rules to assign a particular answer target during the answer typing phase. In this way, they take advantage of the structure inherent in how-questions rather than just treating them as general queries. However, manually hand-crafting types is costly, and would have to be repeated if the system was moved to a new language or a new query domain. Our automatic approach does not suffer from this drawback.

Light et al. (2001) showed that for a small fixed set of answer types, multiple words tagged with the same type will exist even with perfect passage retrieval, sentence retrieval, and type assignment. For example, Example 1 may be answered with a sentence such as "bears range in weight from the smaller black bear at 400 pounds to the gigantic grizzly at over 1200 pounds" in which two answers have appropriate units but only one of which is correct. We provide results in Section 6 confirming the limits of answer typing at narrowing answer focus, using varying levels of perfect information.

Our approach makes use of the web as a large corpus of useful information. Exploiting the vast amount of data on the web is part of a growing trend in Natural Language Processing (Keller and Lapata, 2003). Indeed, many QA systems have been developed using the web (to varying degrees) to assist in finding a correct answer (Brill et al., 2001; Cucerzan and Agichtein, 2005; Radev et al., 2001), as the web is the largest available corpus even if its information can be difficult to harness. Rather than relying on the web to find the answer to a question, we rely on it as a source of information on appropriate units only. Should the domain of the question answering system change from general factoid questions, units may be extracted from a smaller, domain-specific corpus.

3 Model Framework

The objective of our model is to create a list of relevant units for an adjective that may be found in a how-question. We wish to create these lists *a priori* and off-line so that they are applicable to future questions. Although the model described here can be applied on-line at the time of question answering, the resources and time required make off-line generation of unit lists the preferred approach.

We wish to automatically learn a mapping $T : A \rightarrow U_A$ in which A is a set of adjectives derived from how-questions and U_A is a set of lists of units associated with these adjectives. For example, an element of this mapping might be:

$$\textit{high} \in A \rightarrow \{\text{feet, meter, foot, inches, ...}\} \in U_A$$

which assigns height measurements to "how high" questions. Inducing this mapping means establishing a connection, or *co-occurrence*, between each adjective a and its units U_a. In the following subsections, we show how to establish this connection.

3.1 Using WordNet for Adjective Expansion

In common documents, such as news articles or web pages, the co-occurrence of an adjective and its units may be unlikely. For example, the co-occurrence between "heavy" and "pounds" may not be as prevalent as the co-occurrence between "weight" and "pounds." We therefore propose using WordNet (Fellbaum, 1998) to expand the how-adjective a to a set of related concepts the adjective may be used to describe. We denote a related concept of a as r. In the above example, "heavy" can be used to describe a "weight." Two useful WordNet relations are the *attribute* relation, in which the adjective is an attribute of the concept, and in cases where no attribute exists, the *derivationally-related* words. "Heavy" is an attribute of "weight" whereas the derivationally-related form is "heaviness," a plausible but less useful concept. Next we describe how the particular co-occurrence of the related concept r and unit u is obtained.

3.2 Using Google to Obtain Counts

We selected the Google search engine as a source of co-occurrence data due to the large number of indexed documents from which co-occurrence counts can be derived. To further enhance the quality of co-occurrence data, we search on the specific phrase "r is measured in" in which r is one of the related concepts of a. This allows for the simultaneous discovery of unknown units and the retrieval of their co-occurrence counts.

Sentences in which the pattern occurs are parsed using Minipar (Lin, 1998b) so that we can obtain the word related to "measured" via the prepositional *in* relation. This allows us to handle sentential constructions that may intervene between "measured" and a meaningful unit. For each unit u that is related to "measured" via *in*, we increment the co-occurrence count $f(u, r)$, thereby collecting frequency counts for each u with r.

The pattern's precision prevents incidental co-occurrence between a related concept and some unit that may occur simply because of the general topic of the document. For example, "size is measured in" matches "Size is measured in gigabytes, and performance is measured in milliseconds". In this example, the co-occurrence count $f(\text{gigabytes}, \text{size})$ would be incremented by one, whereas there is no co-occurrence between "size" and "milliseconds." Due to the large amount of data available to Google, we can afford to restrict ourselves to a single pattern and still expect to find meaningful units.

To gather the co-occurrence counts between an adjective a and a unit u, we first expand a to the set of related concepts R_a and then compute:

$$f(u, a) = \sum_{r \in R_a} f(u, r) \tag{1}$$

These frequencies can then be used by the scoring functions described in the following section.

3.3 Filtering the Unit List

For a given adjective a and a particular unit u with co-occurrence $f(u, a)$, we define two important statistics:

$$P(u|a) = \frac{f(u,a)}{\sum_{u' \in U} f(u', a)} \quad (2)$$

$$P(a|u) = \frac{f(u,a)}{\sum_{a' \in A} f(u, a')} \quad (3)$$

The first equation measures the likelihood of a unit u being an answer unit for a how-question with the given adjective a. The second equation measures, for a given unit u, how likely a how question with adjective a asked the question answered by u. The second measure is particularly useful in cases where a unit u co-occurs with a number of different adjectives. These units are inherently less useful for answer typing. For example, if the word "terms" occurs on the unit list of adjectives such as "high," "long," and "heavy," it may indicate that "terms" is not an appropriate measure for any of these concepts, but rather just a word likely to co-occur with nouns that can be measured.

We propose using the measures $P(u|a)$ and $P(u|a)P(a|u)$ to score and rank our how-adjective unit lists. $P(a|u)$ alone showed inferior performance on the development set and so will not be further considered. $P(u|a)P(a|u)$ approximates the standard tf-idf measure (Salton and Buckley, 1988). $P(u|a)$ is the term frequency tf in the unit list and $P(a|u)$ is the inverse document frequency idf of the unit over all unit lists. Using these measures, we can create a unit list for an adjective a as

$$U_a = \{u : Score(u, a) \geq \mathcal{T}\} \quad (4)$$

in which $Score(u, a)$ is the score of unit u with adjective a (either $P(u|a)$ or $P(u|a)P(a|u)$) and $\mathcal{T}$ is some threshold imposed to deal with the amount of noise present in the co-occurrence data. This threshold allows us to vary the required strength of the association between the unit and the question in order to consider the unit as appropriate to the how-adjective. In Section 6, we demonstrate this flexibility by showing how answer identification precision and recall can be traded off as desired by the given application. The value $Score(u, a)$ can also be passed to downstream modules of the question answering process (such as the answer extractor), which may then exploit the association value directly.

4 Implementation Details

4.1 Automatic How-Adjective Discovery

An initial step in implementing answer typing for how-adjective questions is to decide which adjectives would benefit from types. WordNet gives a set of all adjectives, but providing answer type units for all these adjectives is unnecessary and potentially misleading. Many adjectives would clearly never occur in a how-adjective query (i.e., "how vehicular...?"), and even some that do, like the "how orwellian" query mentioned above, are difficult to quantify. For these, a simple search with keyword matching as in typical information retrieval would be preferable.

We have a two-stage process for identifying unit-typable how-adjectives. First, we examine the AOL query data (Pass et al., 2006) and extract as candidates all 152 adjectives that occur with the pattern "how *adjective* is/are/was/were." Second, we filter adjectives that do not have a related concept in WordNet (Section 3.1). We built unit lists for the 104 adjectives that remained.

Given that both the query database and WordNet may lack information, it is important to consider the coverage of actual how-adjective questions that unit lists collected this way may have. Reassuringly, experiments have shown 100% coverage of the 96 adjectives in our development and test question set, taken from the TREC QA corpus (see Section 5).

4.2 Similar Word Expansion

Unfortunately, we found that search results obtained using the pattern described in Section 3.2 do not produce a wide variety of units. Web pages often do not use a slang term when mentioning the unit of measurement; a search for "size is measured in gigs" on Google returns zero pages. Also, searching with Google's API and obtaining relevant documents can be time consuming, and we must limit the number of pages considered. Thus, there is strong motivation to expand the list of units obtained from Google by automatically considering *similar* units.

We gather similar units from an automatically-constructed thesaurus of distributionally similar words (Lin, 1998a). The similar word expansion can add a term like *gigs* as a unit for *size* by virtue of its association with *gigabytes*, which is on the original list.

Unit similarity can be thought of as a mapping $S : U \rightarrow V_U$ in which U is a set of units and V_U is sets of related units. If u is an element of U_a for a particular adjective a, the mapping $u \rightarrow V_u$ gives us a way to add new words to the unit list for a. For example, the similar word list for "gigabytes" might be $\{$*GB, megabytes, kilobytes, KB, byte, GHz, gigs*... $\}$, which can all be added to the unit list for the adjective "large."

After expanding each element of the unit list U_a for adjective a, we have a new set of units W_a:

$$W_a = U_a \cup S(U_a) \qquad (5)$$

where $S(U_a) = \bigcup_{u \in U_a} V_u$.

For each $v \in V_u$ there is an associated score $\alpha(u, v) \in [0, 1]$ that measures how similar v is to u. We define the score of units that do not co-occur on similar word lists to be zero and the similarity of two identical units to be one. We can then use these scores to assign estimated co-occurrence counts for any unit w in the expanded unit list W_a:

$$\hat{f}(w, a) = \sum_{u \in U_a} \alpha(w, u) f(u, a) \qquad (6)$$

If a unit $u \in U_a$ also occurs in the set of expanded similar units for another another unit $u' \in U_a$, that is, $u \in V_{u'}$, then the original co-occurrence frequency of u and a, $\alpha(u, u)f(u, a)$, will be boosted by the similarity-weighted frequency of u on the expanded unit list of u', $\alpha(u, u')f(u', a)$.

4.3 Selection of Answer Candidates

For a given how-adjective question and a document of interest, we use a two-stage process to identify the entities in the document that are suitable answers for the question. First, the named entity recognizer of Minipar is used to identify all numerical entities in text, labeled as *NUM*. Minipar labels times, dates, monetary amounts, and address numbers with types other than *NUM* and so we can correctly exclude these from consideration. We then inspect the context of all *NUM* entities to see if a unit exists on the pre-computed unit list for the given how-adjective. Textual entities that pass both stages of our identification process are considered as candidate answers.

5 Experiments

This section presents experiments comparing our how-adjective answer typing approach to alternative schemes on an answer identification task. We compare our two unit ranking functions $P(u|a)$ and $P(u|a)P(a|u)$ (Section 3.3) and test the merits of using the similar unit expansion (Section 4.2).

5.1 Evaluation Questions

The clearest way to test a QA system is to evaluate it on a large set of questions. Although our answer typing system is not capable of fully answering questions, we will make use of the how-adjective questions from TREC 2002-2005 (Vorhees, 2002) as a set of test data. We take eight of the questions as a development set (used for preliminary investigations of scoring functions – no parameters can be set on the development set specifically) and 86 of the questions as a final, unseen test set. Seventeen different adjectives occur in the test questions.

5.2 Evaluation Methodology

We evaluate our system with an approach we call Answer-Identification Precision Recall (AIPR). For a particular scoring threshold (Section 3.3), each adjective has a corresponding unit list, which is used to extract answer candidates from documents (Section 4.3). To ensure the performance of the IR-engine is not an issue in evaluation, we only use documents judged to contain the correct answer by TREC.

Answer-identification precision corresponds to the number of correct answers among the candidate answers extracted by our system. Answer-identification recall is the number of correct answers extracted among the total number of correct answers in the answer documents.

A plot of AIPR allows the designer of a particular QA system to decide on the optimum PR-tradeoff for the answer typing task. If other stages of QA rely on a large number of candidates, a high recall value may be desired so no potential answers are missed. If answer typing is used as a means of boosting already-likely answers, high precision may instead be favoured.

5.3 Comparison Systems

This section describes the various systems we compare with our approach. Recall that perfect AIPR performance is not possible with typing alone (Section 2, (Light et al., 2001)), and thus we provide some of our comparison systems with varying amounts of perfect answer information in order to establish the highest performance possible in different scenarios on the given dataset.

Query-specific Oracle: The best possible system creates a unit list for each specific how-question individually. This list is created using only those units in the answer pattern of the TREC-provided judgement for this specific question.

Adjective-specific Oracle: This system is designed, like ours, to provide a unit list for each how-adjective, rather than for a specific question. The unit list for a particular adjective contains all the units from all the test set answers of how-adjective questions containing that adjective. It is optimal in the sense it will identify every correct answer for each how-adjective, but only contains those units necessary for this identification.

Fixed-Category: This system gives the performance of a general-purpose, fixed-category answer typing approach applied to how-questions. In a fixed-category approach, all how-questions are classified as seeking numerical answers, and thus all numerical answers are returned as answer candidates.

IR-Document Inferred: Here we infer question units from documents believed to be relevant to the question. An IR system (TREC's PRISE) is given a how-adjective question, and returns a set of documents for that query. Every numerical digit in the documents can be considered a possible answer to the question, and the units associated with those values can be collected as the unit list, ranked (and thresholded) by frequency. We remove units that occur in a list of 527 stopwords, and filter numerical modifiers like "hundred, thousand, million, etc."

Answer-Document Inferred: This approach is identical to the IR-Document Inferred approach, except the documents are only those documents judged by TREC to contain the answer. In this way the Answer-Document Inferred approach provides somewhat of an upper bound on Document Inferred unit typing, by assuming perfect document retrieval.

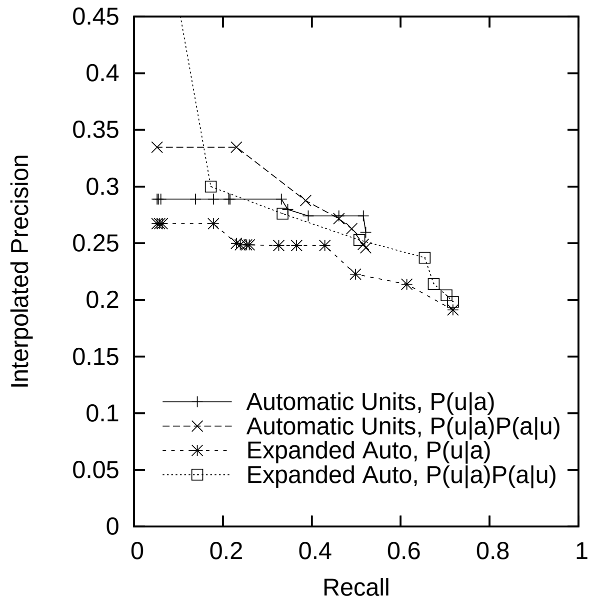


Figure 1: Microaveraged AIPR with different scoring functions, unit lists.

Inferring the answer units from the set of relevant documents is similar in spirit to (Daumé III and Marcu, 2006). In one of their experiments in query-focused summarization, they show competitive summarization performance *without even providing the query,* as the query model is inferred solely from the commonality in relevant documents. In our case, good performance will also be possible if the actual answers have the highest commonality among the numerical values in the relevant documents.

6 Results

The microaveraged Answer-Identification Precision Recall over all question-answer pairs is plotted in Figures 1 and 2. Macroaveraged results are similar.

For our own automatic answer typing approaches, our first observation is the benefit of ranking with $P(u|a)P(a|u)$ as opposed to $P(u|a)$ (Figure 1). Over most of the recall range, both the unexpanded (automatic) unit lists and the expanded unit lists improve in precision by a few percent when using both probabilistic scoring statistics. Secondly, note that both systems using the expanded unit lists can achieve almost 20% higher maximum recall than the unexpanded unit list systems. This provides strong justification for the small overhead of looking up similar words for items on our unit list.

We next examine the AIPR performance of our comparison systems versus our best-performing automatic unit typing approach (Figure 2). The query-specific oracle is able to achieve the highest performance because of perfect knowledge of the units appropriate to a given question. However, its precision is only 42.2%. That is, the answer identification accuracy is limited because the correct answer shares its units with other numerical entities in the answer documents. Slightly worse, the adjective-specific oracle is limited to 34.2% precision. Unlike the query-specific oracle, if the question is "how long did WWII last?", the entities with the irrelevant units "meters" and "kilometers" must also be proposed as candidate answers because they occur in answers to other "how long" questions. This oracle thus provides a more appropriate upper bound on automatic unit-typing performance as our automatic approaches also build unit lists for adjectives rather than questions. Note again that unit lists for adjectives can be generated off-line whereas unit lists for specific questions need the query before processing.

In terms of recall, both upper-bound systems top out at around 78% (with our expanded systems reaching close to this at about 72%). At first, this number seems fairly disappointing: if how-adjective questions only have answer units in 78% of the cases, perhaps our typing approach is not entirely appropriate. On inspecting the actual misses, however, we find that 10 of the 16 missed questions correspond to "how old" questions. These are often answered without units (e.g. "at age 52."). Higher recall would be possible if the system defaults to extracting all numerical entities for "how old" questions. On the remaining questions, high recall can indeed be obtained.

Also of note is the clear disadvantage of using the standard fixed-category approach to how-question answer typing (Figure 2). Its precision runs at just under 5%, about a quarter of the lowest precision of any of our unit-list approaches at any recall value. However, fixed-category typing does achieve high recall, roughly 96%, missing only numerical entities unrecognized by Minipar. This high recall is possible because fixed-category typing does not miss answers for "how old" questions.

Both inferred approaches also perform worse than

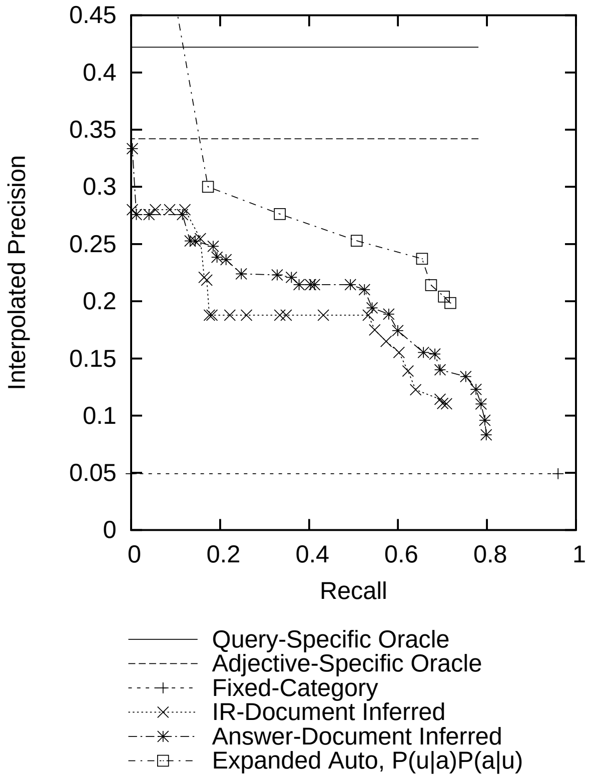


Figure 2: Microaveraged AIPR of our approach versus comparison systems.

our system (Figure 2). Thus inferring units from relevant documents does not seem promising, as even the unrealistic approach of inferring only from known answer documents cannot achieve as high in answer-identification precision. Also, realistically using IR-retrieved documents has universally lower AIPR. As expected, answer-document inferred recall plateaus at the same spot as the oracle systems, as it also requires a unit after each numerical entity (hurting it, again, on the "how old" questions). Despite their lower performance, note that these inferred approaches are completely orthogonal to our offline automatic answer-typing, so a future possibility remains to combine both kinds of systems within a unified framework.

7 Conclusions and Future Work

Although it is difficult to evaluate the impact of our approach until it is integrated into a full QA-system, we have clearly demonstrated the advantages of automatic answer typing for how-questions. We have shown the improvements possible by ranking with different co-occurrence statistics, and the benefit of expanding unit lists with similar words. Experimental results show our approaches achieve superior AIPR performance over all realistic baselines.

In addition to proposing a competitive system, we believe we have established a framework and evaluation methodology that may be of use to other researchers. For example, although manual typing remains an option, our approach can at least provide a good set of candidate units to consider. Furthermore, a similar-word database can expand the list obtained by manual typing. Finally, users may also wish to rank the manual types in some way, and thus configure the system for a particular level of answer-identification precision/recall.

Our success with these unit lists has encouraged two main directions of future work. First, we plan to move to a discriminative approach to combining scores and weighting unit features using a small labeled set. Secondly, we will look at incorporating units into the information retrieval process. Our motivating example in Section 1 retrieved irrelevant documents when given to a search engine, and this seems to be a general trend in how-question IR. Less than 60% of the TREC how-questions have a unit of the correct type anywhere in the top ten documents returned by the PRISE IR engine, and less than half correspondingly had a correct answer in the top ten at all. Making the information retrieval process aware of the desired answer types will be an important future direction of QA research.

Acknowledgments

We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, and the Alberta Informatics Circle of Research Excellence.

References

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. In *TREC*.

S. Cucerzan and E. Agichtein. 2005. Factoid Question Answering over Unstructured and Structured Web Content. In *TREC*.

H. Daumé III and D. Marcu. 2006. Bayesian query-focused summarization. In *COLING-ACL*, pages 305–312.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang. 2005. Employing Two Question Answering Systems in TREC-2005. In *TREC*.

A. Ittycheriah, M. Franz, W-J. Zhu, A. Ratnaparkhi, and R. Mammone. 2000. IBM's Statistical Question Answering System. In *TREC*.

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

X. Li and D. Roth. 2002. Learning Question Classifi ers. In *COLING*, pages 556–562.

M. Light, G. Mann, E. Riloff, and E. Breck. 2001. Analyses for Elucidating Current Question Answering Technology. *Natural Language Engineering*, 7(4):325–342.

D. Lin. 1998a. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–773.

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

G. Pass, A. Chowdhury, and C. Torgeson. 2006. A picture of search. In *The First International Conference on Scalable Information Systems*.

C. Pinchak and D. Lin. 2006. A Probabilistic Answer Type Model. In *EACL*.

J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. 2003. IBM's PIQUANT in TREC2003. In *TREC*.

D. Radev, H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan, and J. Prager. 2001. Mining the Web for Answers to Natural Language Questions. In *CIKM*.

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

E. Vorhees. 2002. Overview of the TREC 2002 question answering track. In *TREC*.

M. Wu, M. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. ILQUA – An IE-Driven Question Answering System. In *TREC*.

A Probabilistic Framework for Answer Selection in Question Answering

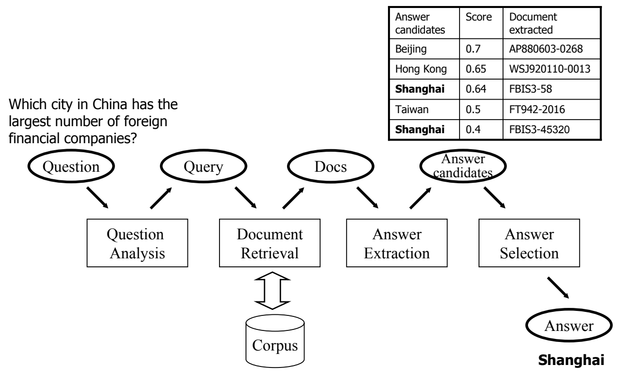
Jeongwoo Ko[1]**, Luo Si**[2]**, Eric Nyberg**[1]
[1]Language Technologies Institute, Carnegie Mellon, Pittsburgh, PA 15213
[2]Department of Computer Science, Purdue University, West Lafayette, IN 47907
`jko@cs.cmu.edu, lsi@cs.purdue.edu, ehn@cs.cmu.edu`

Abstract

This paper describes a probabilistic answer selection framework for question answering. In contrast with previous work using individual resources such as ontologies and the Web to validate answer candidates, our work focuses on developing a unified framework that not only uses multiple resources for validating answer candidates, but also considers evidence of similarity among answer candidates in order to boost the ranking of the correct answer. This framework has been used to select answers from candidates generated by four different answer extraction methods. An extensive set of empirical results based on TREC factoid questions demonstrates the effectiveness of the unified framework.

1 Introduction

Question answering aims at finding exact answers to a user's natural language question from a large collection of documents. Most QA systems combine information retrieval with extraction techniques to identify a set of likely candidates and then utilize some selection strategy to generate the final answers (Prager et al., 2000; Clarke et al., 2001; Harabagiu et al., 2001). Since answer extractors may be based on imprecise empirical methods, the selection process can be very challenging, as it often entails identifying correct answer(s) amongst many incorrect ones.



Answer candidates	Score	Document extracted
Beijing	0.7	AP880603-0268
Hong Kong	0.65	WSJ920110-0013
Shanghai	0.64	FBIS3-58
Taiwan	0.5	FT942-2016
Shanghai	0.4	FBIS3-45320

Figure 1: A traditional QA pipeline architecture

Figure 1 shows a traditional QA architecture with an example question. Given the question "*Which city in China has the largest number of foreign financial companies?*", the answer extraction component produces a ranked list of five answer candidates. Due to imprecision in answer extraction, an incorrect answer ("Beijing") was ranked at the top position. The correct answer ("Shanghai") was extracted from two documents with different confidence scores and ranked at the third and the fifth positions. In order to select "Shanghai" as the final answer, we need to address two issues:

- *Answer Validation*. How do we identify correct answer(s) amongst incorrect ones? Validating an answer may involve searching for facts in a knowledge base, e.g. `IS-A(Shanghai, city)`, `IS-IN(Shanghai, China)`.
- *Answer Similarity*. How do we exploit evidence of similarity among answer candidates?

Proceedings of NAACL HLT 2007, pages 524–531,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

For example, when there are redundant answers ("Shanghai", as above) or several answers which represent a single instance (e.g. "Clinton, Bill" and "William Jefferson Clinton") in the candidate list, how much should we boost the answer candidate scores?

To address the first issue, several answer selection approaches have used semantic resources. One of the most common approaches relies on WordNet, CYC and gazetteers for answer validation or answer reranking; answer candidates are pruned or discounted if they are not found within a resource's hierarchy corresponding to the expected answer type (Xu et al., 2003; Moldovan et al., 2003; Prager et al., 2004). In addition, the Web has been used for answer reranking by exploiting search engine results produced by queries containing the answer candidate and question keywords (Magnini et al., 2002), and Wikipedia's structured information has been used for answer type checking (Buscaldi and Rosso, 2006).

To use more than one resource for answer type checking of location questions, Schlobach et al. (2004) combined WordNet with geographical databases. However, in their experiments the combination actually hurt performance because of the increased semantic ambiguity that accompanies broader coverage of location names. This demonstrates that the method used to combine potential answers may matter as much as the choice of resources.

To address the second issue we must determine how to detect and exploit answer similarity. As answer candidates are extracted from different documents, they may contain identical, similar or complementary text snippets. For example, the United States may be represented by the strings "U.S.", "United States" or "USA" in different documents. It is important to detect this type of similarity and exploit it to boost answer confidence, especially for list questions that require a set of unique answers. One approach is to incorporate answer clustering (Kwok et al., 2001; Nyberg et al., 2003; Jijkoun et al., 2006). For example, we might merge "April 1912" and "14 Apr 1912" into a cluster and then choose one answer as the cluster head. However, clustering raises new issues: how to choose the cluster head and how to calculate the scores of the clustered answers.

Although many QA systems individually address these issues in answer selection, there has been little research on generating a generalized probabilistic framework that allows any validation and similarity features to be easily incorporated.

In this paper we describe a probabilistic answer selection framework to address the two issues. The framework uses logistic regression to estimate the probability that an answer candidate is correct given multiple answer validation features and answer similarity features. Experimental results on TREC factoid questions (Voorhees, 2004) show that our framework significantly improved answer selection performance for four different extraction techniques, when compared to default selection using the individual candidate scores produced by each extractor.

This paper is organized as follows: Section 2 describes our answer selection framework and Section 3 lists the features that generate similarity and validity scores for factoid questions. In Section 4, we describe the experimental methodology and the results. Section 5 describes how we intend to extend our framework to handle complex questions. Finally Section 6 concludes with suggestions for future research.

2 Method

Answer validation is based on an estimate of the probability $P(correct(A_i)|A_i, Q)$, where Q is a question and A_i is an answer candidate to the question. Answer similarity is is based on an estimate of the probability $P(correct(A_i)|A_i, A_j)$, where A_j is similar to A_i. Since both probabilities influence answer selection performance, it is important to combine them in a unified framework and estimate the probability of an answer candidate as: $P(correct(A_i)|Q, A_1, ..., A_n)$.

In this paper, we propose a probabilistic framework that directly estimates $P(correct(A_i)|Q, A_1, ..., A_n)$ using multiple answer validation features and answer similarity features. The framework was implemented with logistic regression, which is a statistical machine learning technique used to predict the probability of a binary variable from input variables. Logistic

$$
\begin{aligned}
&P(correct(A_i)|Q, A_1, ..., A_n) \qquad (1)\\
&\quad \approx P(correct(A_i)|val_1(A_i), ..., val_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i))\\
&\quad = \frac{exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k val_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i))}{1 + exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k val_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i))}\\
&\quad where, sim_k(A_i) = \sum_{j=1(j\neq i)}^{N} sim'_k(A_i, A_j).
\end{aligned}
$$

$$
\vec{\alpha}, \vec{\beta}, \vec{\lambda} = \underset{\vec{\alpha},\vec{\beta},\vec{\lambda}}{\operatorname{argmax}} \sum_{j=1}^{R} \sum_{i=1}^{N_j} log P(correct(A_i)|val_1(A_i), ..., val_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i)) \qquad (2)
$$

regression has been successfully employed in many applications including multilingual document merging (Si and Callan, 2005). In our previous work (Ko et al., 2006), we showed that logistic regression performed well in merging three resources to validate answers to location and proper name questions. We extended this approach to combine multiple similarity features with multiple answer validation features. The extended framework estimates the probability that an answer candidate is correct given the degree of answer correctness and the amount of supporting evidence provided in a set of answer candidates (Equation 1).

In Equation 1, each val$_k$(A$_i$) is a feature function used to produce an answer validity score for an answer candidate A$_i$. Each sim$'_k(A_i, A_j)$ is a similarity function used to calculate an answer similarity between A$_i$ and A$_j$. K1 and K2 are the number of answer validation and answer similarity features, respectively. N is the number of answer candidates.

To incorporate multiple similarity features, each sim$_k(A_i)$ is obtained from an individual similarity metric. For example, if Levenshtein distance is used as one similarity metric, sim$_k(A_i)$ is calculated by summing N-1 Levenshtein distances between one answer candidate and all other candidates. As some string similarity metrics (e.g. Levenshtein distance) produce a number between 0 and 1 (where 1 means two strings are identical and 0 means they are different), similarity scores less than some threshold value are ignored.

The parameters α, β, λ were estimated from training data by maximizing the log likelihood as shown in Equation 2, where R is the number of training questions and N$_j$ is the number of answer candidates for each question Q$_j$. For parameter estimation, we used the Quasi-Newton algorithm (Minka, 2003).

To select correct answers, the initial answer candidate set is reranked according to the estimated probability of each candidate. For factoid questions, the top answer is selected as the final answer to the question. As logistic regression can be used for binary classification with a default threshold of 0.5, we can also use the framework to classify incorrect answers: if the probability of an answer candidate is lower than 0.5, it is considered to be a wrong answer and is filtered out of the answer list. This is useful in deciding whether or not a valid answer exists in the corpus, an important aspect of the TREC QA evaluation (Voorhees, 2004).

3 Feature Representation

This section details the features used to generate answer validity scores and answer similarity scores for our answer selection framework.

3.1 Answer Validation Features

Each answer validation feature produces a validity score which predicts whether or not an answer candidate is a correct answer for the question. This task can be done by exploiting external QA resources such as the Web, databases, and ontologies. For factoid questions, we used gazetteers and WordNet in a knowledge-based approach; we also used Wikipedia and Google in a data-driven approach.

3.1.1 Knowledge-based Features

In order to generate answer validity scores using gazetteers and WordNet, we reused the algorithms described in our previous work (Ko et al., 2006).

Gazetteers: Gazetteers provide geographic information, which allows us to identify strings as instances of countries, their cities, continents, capitals, etc. For answer selection, we used three gazetteer resources: the Tipster Gazetteer, the CIA World Factbook (https://www.cia.gov/cia/publications/factbook/index.html) and information about the US states provided by 50states.com (http://www.50states.com). These resources were used to assign an answer validity score between -1 and 1 to each candidate (Figure 2). A score of 0 means the gazetteers did not contribute to the answer selection process for that candidate. For some numeric questions, range checking was added to validate numeric questions similarly to Prager et al. (2004). For example, given the question *"How many people live in Chile?"*, if an answer candidate is within $\pm$ 10% of the population stated in the CIA World Factbook, it receives a score of 1.0. If it is in the range of 20%, its score is 0.5. If it significantly differs by more than 20%, it receives a score of -1.0. The threshold may vary based on when the document was written and when the census was taken[1].

WordNet: The WordNet lexical database includes English words organized in synonym sets, called *synsets* (Fellbaum, 1998). We used WordNet in order to produce an answer validity score between -1 and 1, following the algorithm in Figure 3. A score

[1]The ranges used here were found to work effectively, but were not explicitly validated or tuned.

1) If the answer candidate directly matches the gazetteer answer for the question, its gazetteer score is **1.0**. (e.g. Given the question "*What continent is Togo on?*", the candidate "*Africa*" receives a score of 1.0.)
2) If the answer candidate occurs in the gazetteer within the subcategory of the expected answer type, its score is **0.5**. (e.g., Given the question "*Which city in China has the largest number of foreign financial companies?*", the candidates "*Shanghai*" and "Boston" receive a score of 0.5 because they are both cities.)
3) If the answer candidate is not the correct semantic type, its score is **-1**. (e.g., Given the question "*Which city in China has the largest number of foreign financial companies?*", the candidate "*Taiwan*" receives a score of -1 because it is not a city.)
4) Otherwise, the score is **0.0**.

Figure 2: Validity scoring with gazetteers.

1) If the answer candidate directly matches WordNet, its WordNet score is **1.0**. (e.g. Given the question "*What is the capital of Uruguay?*", the candidate "*Montevideo*" receives a score of 1.0.)
2) If the answer candidate's hypernyms include a subcategory of the expected answer type, its score is **0.5**. (e.g., Given the question "*Who wrote the book 'Song of Solomon'?*", the candidate "*Mark Twain*" receives a score of 0.5 because its hypernyms include "*writer*".)
3) If the answer candidate is not the correct semantic type, this candidate receives a score of **-1**. (e.g., Given the question "*What state is Niagara Falls located in?*", the candidate "*Toronto*" gets a score of -1 because it is not a state.)
4) Otherwise, the score is **0.0**.

Figure 3: Validity scoring with WordNet.

of 0 means that WordNet does not contribute to the answer selection process for a candidate.

3.1.2 Data-driven Features

Wikipedia and Google were used in a data-driven approach to generate answer validity scores.

Wikipedia: Wikipedia (http://www.wikipedia.org) is a multilingual free on-line encyclopedia. Figure 4 shows the algorithm used to generate an answer validity score from Wikipedia. If there is a Wikipedia document whose title matches an answer candidate, the document is analyzed to obtain the term frequency (tf) and the inverse term

```
For each answer candidate $A_i$,
  1. Initialize the Wikipedia score: $ws(A_i) = 0$
  2. Search for a Wikipedia document whose title is $A_i$
  3. If a document is found, calculate tf.idf score of $A_i$ in the
     retrieved Wikipedia document
       $ws(A_i) = (1+\log(tf)) \times (1+\log(idf))$
  4. If not, for each question keyword $K_j$ ,
    4.1. Search for a Wikipedia document that includes $K_j$
    4.2. If a document is found, calculate tf.idf score of $A_i$
       $ws(A_i) += (1+\log(tf)) \times (1+\log(idf))$
```

Figure 4: Validity scoring with Wikipedia

```
For each answer candidate $A_i$,
  1. Initialize the Google score: $gs(A_i) = 0$
  2. For each snippet $s$:
    2.1. Initialize the snippet co-occurrence score: $cs(s) = 1$
    2.2. For each question keyword $k$ in $s$:
      2.2.1 Compute distance $d$, the minimum number of
        words between $k$ and the answer candidate
      2.2.2 Update the snippet co-occurrence score:
        $cs(s) = cs(s) \times 2^{(1+d)^{-1}}$
    2.3. $gs(A_i) = gs(A_i) + cs(s)$
  3. Normalize the Google score (dividing it by a constant $C$ )
```

Figure 5: Validity scoring with Google

frequency (idf) of the candidate, from which a tf.idf score is calculated. When there is no matched document, each question keyword is also processed as a back-off strategy, and the answer validity score is calculated by summing the tf.idf scores. To calculate word frequency, the TREC Web Corpus (http://ir.dcs.gla.ac.uk/test_collections/wt10g.html) was used as a large background corpus.

Google: Following Magnini et al. (2002), we used Google to generate a numeric score. A query consisting of an answer candidate and question keywords was sent to the Google search engine. To calculate a score, the top 10 text snippets returned by Google were then analyzed using the algorithm in Figure 5.

3.2 Answer Similarity Features

We calculate the similarity between two answer candidates using multiple string distance metrics and a list of synonyms.

3.2.1 String Distance Metrics

There are several different string distance metrics to calculate the similarity of short strings. We used five popular string distance metrics: Levenshtein, Jaccard, Jaro, Jaro-Winkler, and Cosine similarity.

3.2.2 Synonyms

Synonyms can be used as another metric to calculate answer similarity. We defined a binary similarity score for synonyms.

$$sim(A_i, A_j) = \begin{cases} 1, & if\ A_i\ is\ a\ synonym\ of\ A_j \\ 0, & otherwise \end{cases}$$

To get a list of synonyms, we used three knowledge bases: WordNet, Wikipedia and the CIA World Factbook. WordNet includes synonyms for English words. Wikipedia redirection is used to obtain another set of synonyms. For example, “Calif.” is redirected to “California” in Wikipedia, and “William Jefferson Clinton” is redirected to “Bill Clinton”.

The CIA World Factbook includes five different names for a country: conventional long form, conventional short form, local long form, local short form and former name. For example, the conventional long form of Egypt is “Arab Republic of Egypt”, the conventional short form is “Egypt”, the local short form is “Misr”, the local long form is “Jumhuriyat Misr al-Arabiyah” and the former name is “United Arab Republic (with Syria)”. All are considered to be synonyms of “Egypt”.

In addition, manually generated rules are used to obtain synonyms for different types of answer candidates (Nyberg et al., 2003):

- Dates are converted into the ISO 8601 date format (YYYY-MM-DD) (e.g., “April 12 1914” and “12th Apr. 1914” are converted into “1914-04-12” and considered as synonyms).
- Temporal expressions are converted into the HH:MM:SS format (e.g., “six thirty five p.m.” and “6:35 pm” are converted into “18:35:xx” and considered as synonyms).
- Numeric expression are converted into scientific notation (e.g, “one million” and “1,000,000” are converted into “1e+06” and considered as synonyms).

- Representative entities are converted into the represented entity when the expected answer type is `COUNTRY` (e.g., "the Egyptian government" is changed to "Egypt" and "Clinton administration" is changed to "U.S.").

4 Experiment

This section describes the experiments we used to evaluate our answer selection framework. The JAVELIN QA system (Nyberg et al., 2006) was used as a testbed for the evaluation.

4.1 Experimental Setup

A total of 1760 factoid questions from the TREC8-12 QA evaluations served as a dataset, with 5-fold cross validation.

To better understand how the performance of our framework varies for different extraction techniques, we tested it with four JAVELIN answer extraction modules: FST, LIGHTv1, LIGHTv2 and SVM (Nyberg et al., 2006). FST is an answer extractor based on finite state transducers that incorporate a set of extraction patterns (both manually-created and generalized patterns). LIGHTv1 is an extractor that selects answer candidates using a non-linear distance heuristic between the keywords and an answer candidate. LIGHTv2 is another extractor based on a different distance heuristic, originally developed as part of a multilingual QA system. SVM is an extractor that uses Support Vector Machines to discriminate between correct and incorrect answers.

Answer selection performance was measured by average accuracy: the number of correct top answers divided by the number of questions where at least one correct answer exists in the candidate list provided by an extractor. The baseline was calculated with the answer candidate scores provided by each individual extractor; the answer with the best extractor score was chosen, and no validation or similarity processing was performed. For Wikipedia, we used a version downloaded in Nov. 2005, which contained 1,811,554 articles.

4.2 Results and Analysis

We first analyzed the average accuracy when using individual validation features. Figure 6 shows the effect of the individual answer validation features on different extraction outputs. The combination of all features significantly improved the performance when compared to answer selection using a single feature. Comparing the data-driven features with the knowledge-based features, the data-driven features (such as Wikipedia and Google) increased performance more than the knowledge-based features (such as gazetteers and WordNet); our intuition is that the knowledge-based features covered fewer questions. The biggest improvement was found with candidates produced by the SVM extractor: a 242% improvement over the baseline. It was mostly because SVM tended to produce several answer candidates with the same or very similar confidence scores, but our framework could select the correct answer among many incorrect ones by exploiting answer validation features.

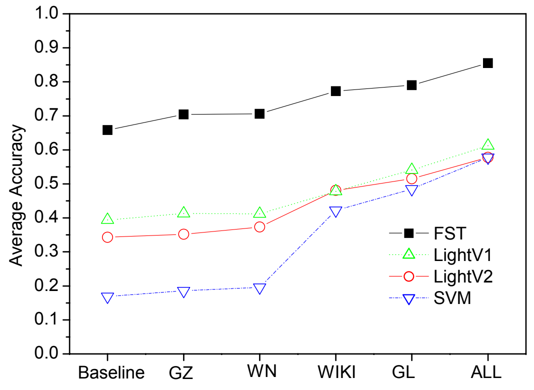


Figure 6: Average accuracy of individual answer validation features (GZ: gazetteers, WN: WordNet, WIKI: Wikipedia, GL: Google, ALL: combination of all features).

Table 1 shows the effect of individual similarity features on different extractors when using 0.3 and 0.5 as a similarity threshold, respectively. When comparing five different string similarity features (Levenshtein, Jaro, Jaro-Winkler, Jaccard and Cosine similarity), Levenshtein and Jaccard tended to perform better than the others. When comparing synonym features with string similarity features, synonyms performed slightly better.

We also analyzed answer selection performance when combining all six similarity features ("All" in Table 1). Combining all similarity features did not improve the performance except for the FST extractor, because including five string similarity features

Similarity feature	FST		LIGHTv1		LIGHTv2		SVM	
	0.3	0.5	0.3	0.5	0.3	0.5	0.3	0.5
Levenshtein	0.728	0.728	**0.471**	0.455	0.399	0.400	0.381	0.383
Jaro	0.708	0.705	0.422	0.440	0.373	0.378	0.274	0.282
Jaro-Winkler	0.701	0.705	0.426	0.442	0.374	0.379	0.277	0.275
Jaccard	0.738	0.738	0.438	0.448	**0.452**	0.448	0.382	0.390
Cosine	0.738	0.738	0.436	0.435	0.418	0.422	0.380	0.378
Synonyms	**0.745**	**0.745**	0.458	0.458	0.442	0.442	**0.412**	**0.412**
Lev+Syn	0.748	**0.751**	0.460	**0.466**	0.445	**0.448**	**0.420**	0.412
Jac+Syn	0.742	0.742	0.456	0.465	0.440	0.445	0.396	0.396
All	0.755	0.755	0.405	0.425	0.435	0.431	0.303	0.302

Table 1: Average accuracy using individual similarity features under different thresholds: 0.3 and 0.5 ("Lev+Syn": the combination of Levenshtein with synonyms, "Jac+Syn": the combination of Jaccard and synonyms, "All": the combination of all similarity metrics)

	Baseline	Sim	Val	All
FST	0.658	0.751	0.855	0.877
LIGHTv1	0.394	0.466	0.612	0.628
LIGHTv2	0.343	0.448	0.578	0.582
SVM	0.169	0.420	0.578	0.586

Table 2: Average accuracy of individual features (Sim: merging similarity features, Val: merging validation features, ALL: combination of all features).

provided too much redundancy to the logistic regression. We also compared the combination of Levenshtein with synonyms and the combination of Jaccard with synonyms, and then chose Levenshtein and synonyms as the two best similarity features in our framework.

We also analyzed the degree to which the average accuracy was affected by answer similarity and validation features. Table 2 compares the average accuracy using the baseline, the answer similarity features, the answer validation features and all feature combinations. As can be seen, the similarity features significantly improved performance, so we can conclude that exploiting answer similarity improves answer selection performance. The validation features also significantly improved the performance.

When combining both sets of features together, the answer selection performance increased for all four extractors: an average of 102% over the baseline, 30% over the similarity features and 1.82% over the validation features. Adding the similarity features to the validation features generated small but consistent improvement in all configurations. We expect more performance gain from similarity features when merging similar answers returned from all four extractors.

5 Extensions for Complex Questions

Although we conducted our experiments on factoid questions, our framework can be easily extended to handle complex questions, which require longer answers representing facts or relations (e.g., *"What is the relationship between Alan Greenspan and Robert Rubin?"*). As answer candidates are long text snippets, different features should be used for answer selection. Possible validation features include question keyword inclusion and predicate structure match (Nyberg et al., 2005). For example, given the question *"Did Egypt sell Scud missiles to Syria?"*, the key predicate from the question is *Sell(Egypt, Syria, Scud missile)*. If there is a sentence which contains the predicate structure *Buy(Syria, Scud missile, Egypt)*, we can calculate the predicate structure distance and use it as a validation feature. For answer similarity, we intend to explore novelty detection approaches evaluated in Allan et al. (2003).

6 Conclusion

In this paper, we described our answer selection framework for estimating the probability that an answer candidate is correct given multiple answer validation and similarity features. We conducted a series of experiments to evaluate the performance of the framework and analyzed the effect of individual validation and similarity features. Empirical results on TREC questions show that our framework improved answer selection performance in the JAVELIN QA system by an average of 102% over the baseline, 30% over the similarity features alone and 1.82% over the validation features alone.

We plan to improve our framework by adding regularization and selecting the final answers among candidates returned from all extractors. As our current framework is based on the assumption that each answer is independent, we are building another probabilistic framework which does not require any independence assumption, and uses an undirected graphical model to estimate the joint probability of all answer candidates.

7 Acknowledgments

This work was supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program award number NBCHC040164.

References

J. Allan, C. Wade, and A. Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of SIGIR*.

D. Buscaldi and P. Rosso. 2006. Mining Knowledge from Wikipedia for the Question Answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*.

C. Clarke, G. Cormack, and T. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of SIGIR*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Grju, V. Rus, and P. Morarescu. 2001. FALCON: Boosting knowledge for answer engines. In *Proceedings of TREC*.

V. Jijkoun, J. van Rantwijk, D. Ahn, E. Tjong Kim Sang, and M. de Rijke. 2006. The University of Amsterdam at CLEF@QA 2006. In *Working Notes CLEF*.

J. Ko, L. Hiyakumoto, and E. Nyberg. 2006. Exploiting semantic resources for answer selection. In *Proceedings of the International Conference on Language Resources and Evaluation*.

C. Kwok, O. Etzioni, and D. S. Weld. 2001. Scaling question answering to the web. In *Proceedings of WWW10 Conference*.

B. Magnini, M. Negri, R. Pervete, and H. Tanev. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA*.

T. Minka. 2003. A Comparison of Numerical Optimizers for Logistic Regression. Unpublished draft.

D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*.

E. Nyberg, T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L. Lita, S. Murtagh, V. Pedro, and D. Svoboda. 2003. The JAVELIN Question-Answering System at TREC 2002. In *Proceedings of the Text REtrieval Conference*.

E. Nyberg, T. Mitamura, R. Frederking, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, V. Pedro, and A. Schlaikjer. 2006. JAVELIN I and II Systems at TREC 2005. In *Proceedings of TREC*.

E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M. Bilotti, A. Schlaikjer, and K. Hannan. 2005. Extending the javelin qa system with domain semantics. In *Proceedings of AAAI-05 Workshop on Question Answering in Restricted Domains*.

J. Prager, E. Brown, A. Coden, and D. Radev. 2000. Question answering by predictive annotation. In *Proceedings of SIGIR*.

J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. 2004 IBM's Piquant in Trec2003. In *Proceedings of TREC*.

S. Schlobach, M. Olsthoorn, and M. de Rijke. 2004. Type checking in open-domain question answering. In *Proceedings of European Conference on Artificial Intelligence*.

L. Si and J. Callan. 2005 CLEF2005: Multilingual retrieval by combining multiple multilingual ranked lists. In *Proceedings of Cross-Language Evaluation Forum*.

E. Voorhees. 2004. Overview of the TREC 2003 question answering track. In *Proceedings of TREC*.

J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. 2003. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of TREC*.

Question Answering using Integrated Information Retrieval and Information Extraction

Barry Schiffman and **Kathleen R. McKeown**
Department of Computer Science
Columbia University
New York, NY 10027
bschiff,kathy@cs.columbia.edu

Ralph Grishman
Department of Computer Science
New York University
New York, NY 10003
grishman@cs.nyu.edu

James Allan
University of Massachusetts
Department of Computer Science
Amherst, MA 01003
allan@cs.umass.edu

Abstract

This paper addresses the task of providing extended responses to questions regarding specialized topics. This task is an amalgam of information retrieval, topical summarization, and Information Extraction (IE). We present an approach which draws on methods from each of these areas, and compare the effectiveness of this approach with a query-focused summarization approach. The two systems are evaluated in the context of the *prosecution* queries like those in the DARPA GALE distillation evaluation.

1 Introduction

As question-answering systems advance from handling factoid questions to more complex requests, they must be able to determine how much information to include while making sure that the information selected is indeed relevant. Unlike factoid questions, there is no clear criterion that defines the kind of phrase that answers the question; instead, there may be many phrases that could make up an answer and it is often unclear in advance, how many. As system developers, our goal is to yield high recall without sacrificing precision.

In response to questions about particular events of interest that can be enumerated in advance, it is possible to perform a deeper semantic analysis focusing on the entities, relations, and sub-events of interest. On the other hand, the deeper analysis may be errorful and will also not always provide complete coverage of the information relevant to the query. The challenge, therefore, is to blend a shallower, robust approach with the deeper approach in an effective way.

In this paper, we show how this can be achieved through a synergistic combination of information retrieval and information extraction. We interleave information retrieval (IR) and response generation, using IR in high precision mode in the first stage to return a small number of documents that are highly likely to be relevant. Information extraction of entities and events within these documents is then used to pinpoint highly relevant sentences and associated words are selected to revise the query for a second pass of retrieval, improving recall. As part of this process, we approximate the relevant context by measuring the proximity of the target name in the query and extracted events.

Our approach has been evaluated in the framework of the DARPA GALE[1] program. One of the GALE evaluations involves responding to questions based on a set of question templates, ranging from broad questions like "Provide information on *X*", where X is an organization, to questions focused on particular classes of events. For the experiments presented here, we used the GALE program's *prosecution* class of questions. These are given in the following form: "Describe the prosecution of *X* for *Y*," where *X* is a person and *Y* is a crime or charge. Our results show that we are able to achieve higher accu-

[1]Global Autonomous Language Exploitation

Proceedings of NAACL HLT 2007, pages 532–539,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

racy with a system that exploits the justice events identified by IE than with an approach based on query-focused summarization alone.

In the following sections, we first describe the task and then review related work in question-answering. Section 3 details our procedure for finding answers as well as performing the information retrieval and information extraction tasks. Section 4 compares the results of the two approaches. Finally, we present our conclusion and plans for future work.

1.1 The Task

The language of the question immediately raises the question of what is meant by prosecution. Unlike a question such as "When was *X* born?", which is expected to be answered by a clear, concrete phrase, the prosecution question asks for a much greater range of material. The answer is in no way limited to the statements and activities of the prosecuting attorney, although these would certainly be part of a comprehensive answer.

In the GALE relevance guidelines[2], the answer can include many facets of the case:

- Descriptions of the accused's involvement in the crime.
- Descriptions of the activities, motivations, and involvement in the crime.
- Descriptions of the person as long as they are related to the trial.
- Information about the defense of the suspect.
- Information about the sentencing of the person.
- Information about similar cases involving the person.
- Information about the arrest of the person and statements made by him or her.
- Reactions of people involved in the trial, as well as statements by officials or reactions by the general public.

[2]BAE Systems Advanced Information Technologies, "Relevance Guidelines for Distillation Evaluation for GALE: Global Autonomous Language Exploitation", Version 2.2, January 25, 2007

The guidelines also provide a catchall instruction to "include reported information believed to be relevant to the case, but deemed inadmissible in a court of law."

It is easy to see that the use of a few search terms alone will be insufficient to locate a comprehensive answer.

We took a broad view of the question type and consider that any information about the investigation, accusation, pursuit, capture, trial and punishment of the individual, whether a person or organization, would be desireable in the answer.

1.2 Overview

The first step in our procedure sends a query tailored to this question type to the IR system to obtain a small number of high-quality documents with which we can determine what name variations are used in the corpus and estimate how many documents contain references to the individual. In the future we will expand the type of information we want to glean from this small set of documents. A secondary search is issued to find additional documents that refer to the individual, or individuals.

Once we have the complete document retrieval, the foundation for finding these types of events lies in the Proteus information extraction component (Grishman et al., 2005). We employ an IE system trained for the tasks of the 2005 Automatic Content Extraction evaluation, which include entity and event extraction. ACE defines a number of general event types, including *justice* events, which cover indictments, accusations, arrests, trials, and sentencings. The union of all these specific categories gives us many of the salient events in a criminal justice case from beginning to end. The program uses the events, as well as the entities, to help identify the passages that respond to the question.

The selection of sentences is based on the assumption that the co-occurrence of the target individual and a judicial event indicates that the target is indeed involved in the event, but these two do not necesssarily occur in the same sentence.

2 Related Work

A large body of work in question-answering has followed from the opening of the Text Retrieval Conference's Q&A track in 1999. The task started as a group of factoid questions and expanded from there into more sophisticated problems. TREC provides a unique testbed of question-answer pairs for researchers and this data has been influential in furthering progress.

In TREC 2006, there was a new secondary task called "complex, interactive Question Answering," (Dang et al., 2006) which is quite close to the GALE problem, though it incorporated interaction to improve results. Questions are posed in a canonical form plus a narrative elaborating on the kind of information requested. An example question (from the TREC guidelines) asks, "What evidence is there for transport of [drugs] from [Bonaire] to the [United States]?" Our task is most similar to the fully-automatic baseline runs of the track, which typically took the form of passage retrieval with query expansion (Oard et al., 2006) or synonym processing (Katz et al., 2006), and not the deeper processing employed in this work.

Within the broader QA task, the *other* question type is closest to the requirements in GALE, but it is too open ended. In TREC, the input for *other* questions is the name or description of the target, and the response is supposed to be all information that did not fit in the answers to the previous questions. While a few GALE questions have similar input, most, including the prosecution questions, provide more detail about the topic in question.

A number of systems have used techniques inspired by information extraction. One of the top systems in the *other* questions category at the 2004 and 2005 evaluations generated lexical-syntactic patterns and semantic patterns (Schone et al., 2004). But they build these patterns from the question. In our task, we took advantage of the structured question format to make use of extensive work on the semantics of selected domains. In this way we hope to determine whether we can obtain better performance by adding more sophisticated knowledge about these domains. The Language Computer Corporation (LCC) has long experimented with incorporating information extraction techniques. Recently, in its system for the *other* type questions at TREC 2005, LCC developed search patterns for 33 target classes (Harabagiu et al., 2005). These patterns were learned with features from WordNet, stemming and named entity recognition.

More and more systems are exploiting the size and redundancy of the Web to help find answers. Some obtain answers from the Web and then project the answer back to the test corpus to find a supporting document (Voorhees and Dang, 2005). LCC used "web boosting features" to add to key words (Harabagiu et al., 2005). Rather than go to the Web and enhance the question terms, we made a beginning at examining the corpus for specific bits of information, in this prototype, to determine alternative realizations of names.

3 Implementation

As stated above, the system takes a query in the XML format required by the GALE program. The query templates allow users to amplify their requests by specifying a timeframe for the information and/or a locale. In addition, there are provisions for entering synonyms or alternate terms for either of the main arguments, i.e. the accused and the crime, and for related but less important terms.

Since this system is a prototype written especially for the GALE evaluation in July 2006, we paid close attention to the way example questions were given, as well as to the evaluation corpus, which consisted of more than 600,000 short news articles. The goal in GALE was to offer comprehensive results to the user, providing all snippets, or segments of texts, that responded to the information request. This required us to develop a strategy that balanced precision against recall. A system that reported only high-confidence answers was in danger of having no answers or far fewer answers than other systems, while a system that allowed lower confidence answers risked producing answers with a great deal of irrelevant material. Another way to look at this balancing act was that it was necessary for a system to know when to quit. For this reason, we sought to obtain a good estimate of the number of documents we wanted to scan for answers.

Answer selection focused first on the name of the suspect, which was always given in the query template. In many of the training cases, the suspect was in the news only because of a criminal charge against him; and in most, the charge specified was the only accusation reported in the news. Both location and date constraints seemed to be largely superfluous, and so we ignored these. But we did have a mechanism for obtaining supplementary answers keyed to the brief description of the crime and other related words

The first step in the process is to request a seed collection of 10 documents from the IR system. This number was established experimentally. The IR query combines terms tailored to the prosecution template and the specific template parameters for a particular question. The 10 documents returned are then examined to produce a list of name variations that substantially match the name as rendered in the query template. The IR system is then asked for the number of times that the name appears in the corpus. This figure is adjusted by the frequency per document in the seed collection and a new query is submitted, set to obtain the *N* documents in which we expect to find the target's name.

3.1 Information Retrieval

The goal of the information retrieval component of the system was to locate relevant documents that the summarization system could then use to construct an answer. All search, whether high-precision or high-recall, was performed using the Indri retrieval system [3] (Strohman et al., 2005).

Indri provides a powerful query language that is used here to combine numerous aspects of the query. The Indri query regarding Saddam Hussein's prosecution for crimes against humanity includes the following components: source restrictions, prosecution-related words, mentions of Saddam Hussein, justice events, dependence model phrases (Metzler and Croft, 2005) regarding the crime, and a location constraint.

The first part of the query located references to prosecutions by looking for the keywords *prosecution*, *defense*, *trial*, *sentence*, *crime*, *guilty*, or *accuse*, all of which were determined on training data to occur in descriptions of prosecutions. These words were important to have in documents for them to be considered relevant, but the individual's name and the description of the crime were far more important (by a factor of almost 19 to 1).

The more heavily weighted part of the query, then, was a "justice event" marker found using information extraction (Section 3.2) and the more detailed description of that event based on phrases extracted from the crime (here *crimes against humanity*). Those phrases give more probability of relevance to documents that use more terms from the crime. It also included a location constraint (here, *Iraq*) that boosted documents referring to that location. And it captured user-provided equivalent words such as *Saddam Hussein* being a synonym for *former President of Iraq*.

The most complex part of the query handled references to the individual. The extraction system had annotated all person names throughout the corpus. We used the IR system to index all names across all documents and used Indri to retrieve any name forms that matched the individual. As a result, we were able to find references to *Saddam*, *Hussein*, and so on. This task could have also been accomplished with cross-document coreference technology but our approach appeared to compensate for incorrectly translated names slightly better than the coreference system we had available at the time. For example, *Present rust Hussein* was one odd form that was matched by our simple approach.

The final query looked like the following:

```
#filreq( #syn( #1(AFA).source ... #1(XIE).source )
   #weight(
     0.05 #combine( prosecution defense trial sentence
                    crime guilty accuse )
     0.95 #combine(
        #any:justice
        #weight(1.0 #combine(humanity against crimes)
               1.0 #combine(
                  #1(against humanity)
                  #1(crimes against)
                  #1(crimes against humanity))
                1.0 #combine
                   #uw8(against humanity)
                   #uw8(crimes humanity)
                   #uw8(crimes against)
                   #uw12(crimes against humanity)))
        Iraq

        #syn( #1(saddam hussein)
              #1(former president iraq))

        #syn( #equals( entity 126180 ) ...))))
```

The actual query is much longer because it contains 100 possible entities and numerous sources. The processing is described in more detail elsewhere (Kumaran and Allan, 2007).

3.2 Information Extraction

The Proteus system produces the full range of annotations as specified for the ACE 2005 evaluation, including entities, values, time expressions, relations,

[3] http://lemurproject.org/indri

and events. We focus here on the two annotations, entities and events, most relevant to our question-answering task. The general performance on entity and event detection in news articles is within a few percentage points of the top-ranking systems from the evaluation.

The extraction engine identifies seven semantic classes of entities mentioned in a document, of which the most frequent are persons, organizations, and GPE's (geo-political entities – roughly, regions with a government). Each entity will have one or more *mentions* in the document; these mentions include names, nouns and noun phrases, and pronouns. Text processing begins with an HMM-based named entity tagger, which identifies and classifies the names in the document. Nominal and pronominal mentions are identified either with a chunker or a full Penn-Treebank parser. A rule-based coreference component identifies coreference relations, forming entities from the mentions. Finally, a semantic classifier assigns a class to each entity based on the type of the first named mention (if the entity includes a named mention) or the head of the first nominal mention (using statistics gathered from the ACE training corpus).

The ACE annotation guidelines specify 33 different event subtypes, organized into 8 major types. One of the major types is justice events, which include arrest, charge, trial, appeal, acquit, convict, sentence, fine, execute, release, pardon, sue, and extradite subtypes. In parallel to entities, the event tagger first identifies individual event mentions and then uses event coreference to form events. For the ACE evaluation, an annotated corpus of approximately 300,000 words is used to train the event tagger.

For each event mention in the corpus, we collect the trigger word (the main word indicating the event) and a pattern recording the path from the trigger to each event argument. These paths are recorded in two forms: as the sequence of heads of maximal constituents between the trigger and the argument, and as the sequence of predicate-argument relations connecting the trigger to the argument[4]. In addition, a set of maximum-entropy classifiers are trained: to distinguish events from non-events, to classify events by type and subtype, to distinguish arguments from non-arguments, and to classify arguments by argument role. In tagging new data, we first match the context of each instance of a trigger word against the collected patterns, thus identifying some arguments. The argument classifier is then used to collect additional arguments within the sentence. Finally, the event classifier (which uses the proposed arguments as features) is used to reject unlikely events. The patterns provide somewhat more precise matching, while the argument classifiers improve recall, yielding a tagger with better performance than either strategy separately.

[4]These predicate argument relations are based on a representation called GLARF (Grammatical-Logical Argument Representation Framework), which incorporates deep syntactic relations and the argument roles from PropBank and NomBank.

3.3 Answer Generation

Once the final batch of documents is received, the answer generator module selects candidate passages. The names, with alternate renderings, are located through the entity mentions by the IE system. All sentences that contain a *justice* event and that fall within a mention of a target by no more than *n* sentences, where *n* is a settable parameter, which was put at 5 for this evaluation, form the core of the system's answer.

The tactic takes the place of topic segmentation, which we used for other question types in GALE that did not have the benefit of the sophisticated event recognition offered by the IE system. Segmentation is used to give users sufficient context in the answer without needing a means of identifying difficult definite nominal resolution cases that are not handled by extraction.

In order to increase recall, in keeping with the need for a comprehensive answer in the GALE evaluation, we added sentences that contain the name of the target in documents that have *justice* events and sentences that contain words describing the crime. However, we imposed a limitation on the growth of the answer size. When the target individual is well-known, he or she will be mentioned in numerous contexts, reducing the likelihood that this additional mention will be relevant. Thus, when the size of the answer grew too rapidly, we stopped including these additional sentences, and produced sentences only from the *justice* events. The threshold for triggering this shift was 200 sentences.

3.4 Summarization

As a state-of-the-art baseline, we used a generic multidocument summarization system that has been tested in numerous contexts. It is, indeed, the backup answer generator for several question types, including the prosecution questions, in our GALE system, and has been been tested in the topic-based tasks of the 2005 and 2006 Document Understanding Conferences.

A topic statement is formed by collapsing the template arguments into one list, e.g., "saddam hussein crimes against humanity prosecution", and the answer generation module proceeds by using a hybrid approach that combines top-down strategies based on syntactic patterns, alongside a suite of summarization methods which guide content in a bottom-up manner that clusters and combines the candidate sentences (Blair-Goldensohn and McKeown, 2006).

4 Evaluation

The results of our evaluation are shown in Table 1. We increased the number of test questions over the number used in the official GALE evaluation and we used only previously unseen questions. Documents for the baseline system were selected without use of the event annotations from Proteus.

We paired the 25 questions for judges, so that both the system's answer and the baseline answer were assigned to the same person. We provided explicit instructions on the handling on implicit references, allowing the judges to use the context of the question and other answer sentences to determine if a sentence was relevant – following the practice of the GALE evaluation.

Our judges were randomly assigned questions and asked whether the snippets, which in our case were individual sentences, were relevant or not; they could respond *Relevant*, *Not Relevant* or *Don't Know*. In cases where references were unclear, the judges were asked to choose *Don't Know* and these were removed from the scoring.[5]

[5]In the GALE evaluation, the snippets are broken down by hand into *nuggets* – discrete pieces of information – and the answers are scored on that basis. However, we scored our responses on the basis of snippets (sentences) only, as it is much more efficient, and therefore more feasible to repeat in the future.

Our system using IE event detection and entity tracking outperformed the summarization-based baseline, with average precision of 68% compared with 57%. Moreover, the specialized system sustained that level of precision although it returned a much larger number of snippets, totaling 2,086 over the 25 questions, compared with 363 for the baseline system. We computed a relative recall score, using the union of the sentences found by the systems and judged relevant as the ground truth. For recall, the specialized system scored an average 89% versus 17% for the baseline system. Computing an F-measure weighting precision and recall equally, the specialized system outperformed the baseline system 75% to 23%. The difference in relative recall and F-measure are both statisticaly significant under a two-tailed, paired t-test, with $p < 0.001$.

5 Conclusion and Future Work

Our results show that the specialized system statistically outperforms the baseline, a well-tested query focused summarization approach, on precision. The specialized system produced a much larger answer on average (Table 1). Moreover, our answer generator seemed to adapt well to information in the corpus. Of the six cases where it returned fewer than 10 sentences, the baseline found no additional sentences four times (Questions B006, B011, B015 and B022). We regard this as an important property in the question-answering task.

A major challenge is to ascertain whether the mention of the target is indeed involved in the recognized *justice* event. Our event recognition system was developed within the ACE program and only seeks to assigns roles within the local context of a single sentence. We currently use a threshold to consider whether an entity mention is reliable, but we will experiment with ways to measure the likelihood that a particular sentence is about the prosecution or some other issue. We are planning to obtain various pieces of information from additional secondary queries to the search engine. Within the GALE program, we are limited to the defined corpus, but in the general case, we could add more varied resources.

In addition, we are working to produce answers using text generation, to bring more sophisticated summarization techniques to make a better presen-

QID	System with IE				Baseline System			
	Precision	Recall	F-meas	Count	Precision	Recall	F-meas	Count
B001	0.728	0.905	0.807	92	0.818	0.122	0.212	11
B002	0.713	0.906	0.798	108	0.889	0.188	0.311	18
B003	0.770	0.942	0.848	148	0.875	0.058	0.109	8
B004	0.930	0.879	0.904	86	1.000	0.154	0.267	14
B005	0.706	0.923	0.800	34	0.400	0.231	0.293	15
B006	1.000	1.000	1.000	3	0.000	0.000	0.000	17
B007	0.507	1.000	0.673	73	0.421	0.216	0.286	19
B008	0.791	0.909	0.846	201	0.889	0.091	0.166	18
B009	0.759	0.960	0.848	158	0.941	0.128	0.225	17
B010	1.000	0.828	0.906	24	0.500	0.276	0.356	16
B011	0.500	1.000	0.667	6	0.000	0.000	0.000	18
B012	0.338	0.714	0.459	74	0.765	0.371	0.500	17
B013	0.375	0.900	0.529	120	0.700	0.280	0.400	20
B014	0.571	0.800	0.667	7	0.062	0.200	0.095	16
B015	0.500	1.000	0.667	2	0.000	0.000	0.000	10
B016	1.000	0.500	0.667	5	0.375	0.600	0.462	16
B017	1.000	1.000	1.000	13	0.125	0.077	0.095	7
B018	0.724	0.993	0.837	199	0.875	0.048	0.092	8
B019	0.617	0.954	0.749	201	0.684	0.100	0.174	19
B020	0.923	0.727	0.814	26	0.800	0.364	0.500	15
B021	0.562	0.968	0.711	162	0.818	0.096	0.171	11
B022	0.667	1.000	0.800	6	0.000	0.000	0.000	18
B023	0.684	0.950	0.795	196	0.778	0.050	0.093	9
B024	0.117	0.636	0.197	60	0.714	0.455	0.556	7
B025	0.610	0.943	0.741	82	0.722	0.245	0.366	18
Aver	**0.684**	**0.893**	**0.749**	**83**	**0.566**	**0.174**	**0.229**	**14**

Table 1: The table compares results of our answer generator combining the Indri and the Proteus ACE system, against the focused-summarization baseline. This experiment is over 25 previously unseen questions. The differences between the two systems are statistically significant ($p < 0.001$) for recall and f-measure by a two-tailed, paired t-test. A big difference between the two systems is that the answer generator produces a total of 2,086 answer sentences while sustaining an average precision of 0.684. In only three cases, does the precision fall below 0.5. In contrast, the baseline system produced only 362, one-sixth the number of answer sentences. While its average precision was not significantly worse than the answer-generator's, its precision varied widely, failing to find any correct sentences four times.

tation than an unordered list of sentences.

Finally, we will look into applying the techniques used here on other topics. The first test would reasonably be *Conflict* events, for which the ACE program has training data. But ultimately, we would like to adapt our system to arbitrary topic areas.

Acknowledgements

This material is based in part upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

References

Sasha Blair-Goldensohn and Kathleen McKeown. 2006. Integrating rhetorical-semantic relation models for query-focused summarization. In *Proceedings of 6th Document Understanding Conference (DUC2006)*.

Hoa Trang Dang, Jimmy Lin, and Diane Kelly. 2006. Overview of the TREC 2006 question answering track. In *Proceedings TREC*. Forthcoming.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's english ACE 2005 system description. In *ACE 05 Evaluation Workshop*. On-line at http://nlp.cs.nyu.edu/publication.

Sanda Harabagiu, Dan Moldovan, Christine Clark, Mitchell Bowden, Andrew Hickl, and Patrick Wang. 2005. Employing two question answering systems in TREC 2005. In *Proceedings of the Fourteenth Text Retrieval Conference*.

B. Katz, G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner, and A. Wilcox. 2006. External knowledge sources for question answering. In *Proceedings of TREC*. On-line at http://www.trec.nist.gov.

Giridhar Kumaran and James Allan. 2007. Information retrieval techniques for templated queries. In *Proceedings of RIAO*. Forthcoming.

D. Metzler and W.B. Croft. 2005. A Markov random field model for term dependencies. In *Proceedings of ACM SIGIR*, pages 472–479.

D. Oard, T. Elsayed, J. Wang, Y. Wu, P. Zhang, E. Abels, J. Lin, and D. Soergel. 2006. Trec 2006 at maryland: Blog, enterprise, legal and QA tracks. In *Proceedings of TREC*. On-line at http://www.trec.nist.gov.

Patrick Schone, Gary Ciany, Paul McNamee, James Mayeld, Tina Bassi, and Anita Kulman. 2004. Question answering with QACTIS at TREC-2004. In *Proceedings of the Thirteenth Text Retrieval Conference*.

T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. 2005. Indri: A language-model based search engine for complex queries (extended version). Technical Report IR-407, CIIR, UMass Amherst.

Ellen M. Voorhees and Hoa Trang Dang. 2005. Overview of the TREC 2005 question answering track. In *Proceedings of the Fourteenth Text Retrieval Conference*.

Toward Multimedia: A String Pattern-based Passage Ranking Model for Video Question Answering

Yu-Chieh Wu
Dept. of Computer Science and Information Engineering
National Central University
Taoyuan, Taiwan
bcbb@db.csie.ncu.edu.tw

Jie-Chi Yang
Graduate Institute of Network Learning Technology
National Central University
Taoyuan, Taiwan
yang@cl.ncu.edu.tw

Abstract

In this paper, we present a new string pattern matching-based passage ranking algorithm for extending traditional text-based QA toward videoQA. Users interact with our videoQA system through natural language questions, while our system returns passage fragments with corresponding video clips as answers. We collect 75.6 hours videos and 253 Chinese questions for evaluation. The experimental results showed that our method outperformed six top-performed ranking models. It is 10.16% better than the second best method (language model) in relatively MRR score and 6.12% in precision rate. Besides, we also show that the use of a trained Chinese word segmentation tool did decrease the overall videoQA performance where most ranking algorithms dropped at least 10% in relatively MRR, precision, and answer pattern recall rates.

1 Introduction

With the drastic growth of video sources, effective indexing and retrieving video contents has recently been addressed. The well-known Informedia project (Wactlar, 2000) and TREC-VID track (Over et al., 2005) are the two famous examples. Although text-based question answering (QA) has become a key research issue in past decade, to support multimedia such as video, it is still beginning.

Over the past five years, several video QA studies had investigated. Lin et al. (2001) presented an earlier work on combining videoOCR and term weighting models. Yang et al. (2003) proposed a complex videoQA approach by employing abundant external knowledge such as, Web, WordNet, shallow parsers, named entity taggers, and human-made rules. They adopted the term-weighting method (Pasca, and Harabagiu, 2001) to rank the video segments by weighting the pre-defined keywords. Cao and Nunamaker (2004) developed a lexical pattern matching-based ranking method for a domain-specific videoQA. In the same year, Wu et al. (2004) designed a cross-language (English-to-Chinese) video question answering system based on extracting pre-defined named entity words in captions. On the other hand, Zhang and Nunamaker (2004) made use of the simple TFIDF term weighting schema to retrieve the manual-segmented clips for video caption word retrieval. They also manually developed the ontology to improve system performance.

In this paper, we present a new string pattern matching-based passage ranking algorithm for video question answering. We consider that the passage is able to answer questions and also suitable for videos because itself forms a very natural unit. Lin et al. (2003) showed that users prefer passage-level answers over short answer phrases since it contains rich context information. Our method makes use of the string pattern searching in the suffix trees to find common subsequences between a passage and question. The proposed term weighting schema is then designed to compute passage score. In addition, to avoid generating over-length subsequence, we also present two algorithms for re-tokenization and weighting.

2 The Framework of our VideoQA System

An overview of the proposed videoQA system can be shown in Figure 1. The video processing component recognizes the input video as an OCR document at the first stage. Second, each three consecutive sentences were grouped into a passage. We tokenized the Chinese words with three grained sizes: unigram, bigram, and trigram. Similarly, the input question is also tokenized to unigram, bigram, and trigram level of words. To reduce most irrelevant passages, we adopted the BM-25 ranking model (Robertson et al., 2000) to retrieve top-1000 passages as the "input passages". Finally, the proposed passage ranking algorithm retrieved top-*N* passages as answers in response to the question. In the following parts, we briefly introduce the employed videoOCR approach. Section 2.2 presents the sentence and passage segmentation schemes. The proposed ranking algorithms will be described in Section 3.

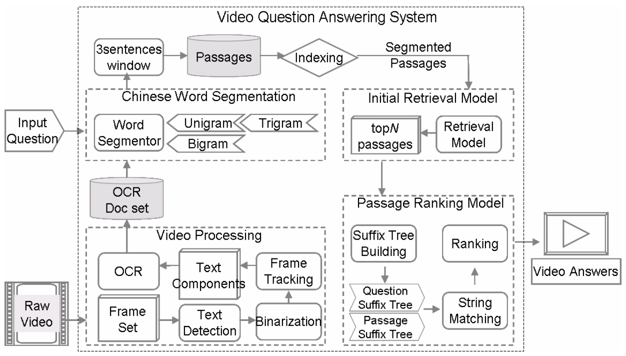


Figure1: System Architecture of the proposed videoQA system

2.1 Video Processing

Our video processing takes a video and recognizes the closed captions as texts. An example of the input and output associated with the whole video processing component can be seen in Figure 2. The videoOCR technique consists of four important steps: text detection, binarization, frame tracking, and OCR. The goal of text detection is to locate the text area precisely. In this paper, we employ the edge-based filtering (Lyu et al., 2005) and slightly modify the coarse-to-fine top-down block segmentation methods (Lienhart and Wernicke, 2002) to find each text component in a frame. The former removes most non-edge areas with global and local thresholding strategy (Fan et al., 2001) while the latter incrementally segments and refines text blocks using horizontal and vertical projection profiles.

The next steps are text binarization and frame tracking. As we know, the main constituent of video is a sequence of image frames. A text component almost appears more than once. To remove redundancy, we count the proportion of overlapping edge pixels between two consecutive frames. If the portion is above 70%, then the two frames were considered as containing the same text components. We then merge the two frames by averaging the gray-intensity for each pixel in the same text component. For the binarization stage, we employ the Lyu's text extraction algorithm (Lyu et al., 2005) to binarize text pixels for the text components. Unlike previous approaches (Lin et al., 2001; Chang et al., 2005), this method does not need to assume the text is in either bright or dark color (but assume the text color is stable). At the end of this step, the output text components are prepared for OCR.

The target of OCR is to identify the binarized text image to the ASCII text. In this paper, we developed a naïve OCR system based on nearest neighbor classification algorithms and clustering techniques (Chang et al., 2005). We also adopted the word re-ranking methods (Lin et al., 2001, strategy 3) to improve the OCR errors.

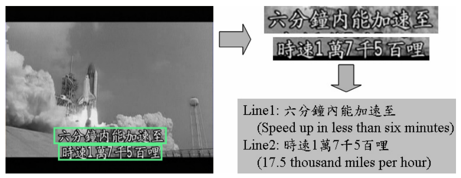


Figure 2: Text extraction results of an input image

2.2 Sentence and Passage Segmentation

In this paper, we treat all words appear in the same frame as a sentence and group every three consecutive sentences as a passage. Usually, words that occur in the same frame provide a sufficient and complete description. We thus consider these words as a sentence unit for sentence segmentation. An example of a sentence can be found in Figure 2. The sentence of this frame is the cascading of the two text lines, i.e. "speed-up to 17.5 thousand miles per hour in less than six minutes" For each OCR document we grouped every three continuous sentences with one previous sentence overlapping to represent a passage. Subsequently, we tokenized Chinese word with unigram, bigram, and trigram levels.

Searching answers in the whole video collection is impractical since most of them are irrelevant to the question. By means of text retrieval technology, the search space can be largely reduced and limited in a small set of relevant document. The document retrieval methods have been developed well and successfully been applied for retrieving relevant passages for question answering (Tellex et al., 2003). We replicated the Okapi BM-25 (Robertson et al., 2000), which is the effective and efficient retrieval algorithms to find the related segmented passages. For each input question, the top-1000 relevant passages are input to our ranking model.

3 The Algorithm

Tellex et al. (2003) compared seven passage retrieval models for text QA except for several adhoc approaches that needed either human-generated patterns or inference ontology which were not available. In their experiments, they showed that the density-based methods (Lee et al., 2001) achieved the best results, while the BM-25 (Robertson, 2000) reached slightly worse retrieval result than the density-based approaches, which adopted named entity taggers, thesaurus, and WordNet. Cui et al. (2005) showed that their fuzzy relation syntactic matching method outperformed the density-based methods. But the limitation is that it required a dependency parser, thesaurus, and training data. In many Asian languages like Chinese, Japanese, parsing is more difficult since it is necessary to resolve the word segmentation problem before part-of-speech (POS) tagging, and parsing (Fung et al., 2004). This does not only make the parsing task harder but also required to train a high-performance word segmentor. The situation is even worse when text contains a number of OCR error words. In addition, to develop a thesaurus and labeled training set for QA is far time-consuming. In comparison to Cui's method, the term weighting-based retrieval models are much less cost, portable and more practical. Furthermore, the OCR document is not like traditional text articles that have been human-typed well where some words were error predicted, unrecognizable, and false-alarm. These unexpected words deeply affect the performance of Chinese word segmentation, and further for parsing. In our experiments (see Table 2 and Table 3), we also showed that the use of a well-trained high-performance Chinese word segmentation tool gave the worse result than using the unigram-level of Chinese word (13.95% and 13.92% relative precision and recall rates dropped for language model method).

To alleviate this problem, we treat the atomic Chinese unigram as word and present a weighted string pattern matching algorithm. Our solution is to integrate the suffix tree for finding, and encoding important subsequence information in trees. Nevertheless, it is known that the suffix tree construction and pattern searching can be accomplished in linear time (Ukkonen, 1995). Before introducing our method, we give the following notations.

passage P = $PW_1, PW_2, \ldots, PW_T$
question Q = $QW_1, QW_2, \ldots, QW_{T'}$
a common subsequence for passage

$$\mathrm{Sub}_i^{\mathrm{P}} = PW_k, PW_{k+1}, \ldots, PW_{k+x-1} \quad \text{if } |\mathrm{Sub}_i^{\mathrm{P}}| = x$$

a common subsequence for question

$$\mathrm{Sub}_j^{\mathrm{Q}} = QW_l, QW_{l+1}, \ldots, QW_{l+y-1} \text{ if } |\mathrm{Sub}_j^{\mathrm{Q}}| = y$$

A common subsequence represents a continuous string matching between P and Q. We further impose two symbols on a subsequence. For example, $\mathrm{Sub}_i^{\mathrm{P}}$ means *i*-th matched continuous string (common subsequence) in the passage, while $\mathrm{Sub}_j^{\mathrm{Q}}$ indicates the *j*-th matched continuous string in the question. The common subsequences can be extracted through the suffix tree building and pattern searching. For example, to extract the set of $\mathrm{Sub}_i^{\mathrm{P}}$, we firstly build the suffix tree of P and incrementally insert substring of Q and label the matched common string between P and Q. Similarly, one can apply a similar approach to generate the set of $\mathrm{Sub}_j^{\mathrm{Q}}$. By extracting all subsequences for P and Q, we then compute the following score (see equation (1)) to rank passages.

$$\begin{aligned}&\text{Passage\_Score(P)} = \lambda \times \text{QW\_Density(Q, P)} + \\ &(1-\lambda) \times \text{QW\_Weight(Q, P)}\end{aligned} \tag{1}$$

The first term of equation (1) "QW_Density(Q, P)" estimates the question word density degree in the passage P, while "QW_Weight(Q, P)" measures the matched question word weights in P. λ is a parameter, which is used to adjust the importance of the QW_Density(Q, P). Both the two estimations make use of the subsequence information for P and Q. In the following parts, we introduce the computation of QW_Density(Q,P) and QW_Weight(Q, P) separately. The time complexity analysis of our method is then discussed in the tail of this section.

The QW_Density(Q, P) is designed for quantifying "how dense the matched question words in the passage P". It also takes the term weight into account. By means of extracting common subsequence in the question, the set of $\mathrm{Sub}_j^{\mathrm{Q}}$ can be used to measures the question word density. At the beginning, we define equation (2) for weighting a subsequence $\mathrm{Sub}_j^{\mathrm{Q}}$.

$$\text{Weight}(\text{Sub}_j^{\text{Q}}) = \text{length}(\text{Sub}_j^{\text{Q}})^{\alpha_1} \times \text{DP}(\text{Sub}_j^{\text{Q}}) \quad (2)$$

Where length(Sub_j^{Q}) is merely the length of Sub_j^{Q} i.e., the number of words in Sub_j^{Q}. α_1 is a parameter that controls the weight of length for Sub_j^{Q}. In this paper, we consider the long subsequence match is useful. A long *N*-gram is usually much less ambiguous than its individual unigram. The second term in equation (2) estimates the "discriminative power" (DP) of the subsequence. Some high-frequent and common words should be given less weight. To measure the DP score, we extend the BM-25 (Robertson et al., 2000) term weighting schema. Equation (3), (4), and (5) list our DP scoring functions.

$$\text{DP}(\text{Sub}_j^{\text{Q}}) = W' \times \frac{(k_1+1)\times \text{TF}(\text{Sub}_j^{\text{Q}},\text{P})}{K+\text{TF}(\text{Sub}_j^{\text{Q}},\text{P})} \times \frac{(k_3+1)\times \text{TF}(\text{Sub}_j^{\text{Q}},\text{Q})}{k_3+\text{TF}(\text{Sub}_j^{\text{Q}},\text{Q})} \quad (3)$$

$$W' = \log\left(\frac{N_P - \text{PF}(\text{Sub}_j^{\text{Q}}) + 0.5}{\text{PF}(\text{Sub}_j^{\text{Q}}) + 0.5}\right) \quad (4)$$

$$K = (1-b) + b \times \frac{|\text{P}|}{\text{AVG}(|\text{P}|)} \quad (5)$$

k_1, b, k_3 are constants, which empirically set as 1.2, 0.75, 500 respectively (Robertson et al., 2000). $\text{TF}(\text{Sub}_j^{\text{Q}},\text{Q})$ and $\text{TF}(\text{Sub}_j^{\text{Q}},\text{P})$ represent the term frequency of Sub_j^{Q} in question Q and passage P. Equation (4) computes the inverse "passage frequency" (PF) of Sub_j^{Q} as against to the traditional inverse "document frequency" (DF) where N_p is the total number of passages. The collected Discovery video is a small but "long" OCR document set, which results the estimation of DF value unreliable. On the contrary, a passage is more coherent than a long document, thus we replace the DF estimation with PF score. It is worth to note that some Sub_j^{Q} might be too long to be further re-tokenized into finer grained size. We therefore propose two algorithms to 1): re-tokenize an input subsequence, and 2): compute the DP score for a subsequence. Figure 3, and Figure 4 list the proposed two algorithms.

The proposed algorithm 1, and 2 can be used to compute and tokenize the DP score of not only Sub_j^{Q} for question but also Sub_j^{P} for passage. As seeing in Figure 4, it requires DP information for different length of *N*-gram. As noted in Section 2.2, the unigram, bigram, and trigram level of words had been stored in indexed files for efficient retrieving and computing DP score at this step. By applying algorithm 1 for the set of Sub_j^{Q}, we can obtain all retokenized subsequences (TSub_j). We then use the re-tokenized subsequences to compute the final density score. Equation (6) lists the QW_Density scoring function.

$$\text{QW\_Density}(\text{Q},\text{P}) = \sum_{i=1}^{T\_CNT-1} \frac{\text{Weight}(\text{TSub}_i) + \text{Weight}(\text{TSub}_{i+1})}{\text{dist}(\text{TSub}_i,\text{TSub}_{i+1})^{\alpha_2}} \quad (6)$$

$$\text{dist}(\text{TSub}_i,\text{TSub}_{i+1}) = \text{min\_distance\_between}(\text{TSub}_i,\text{TSub}_{i+1})\text{\_in\_P} + 1 \quad (7)$$

T_CNT is the total number of retokenized subsequences in Q, which can be extracted through applying algorithm 1 for all Sub_j^{Q}. Equation (7) merely counts the minimum number of words between two neighboring TSub_i, and TSub_{i+1} in the passage. α_2 is the parameter that controls the impact of distance measurement.

```
Algorithm 1: Retokenizing_a_subsequence
Input:
    A subsequence $Sub_j^Q$ where $start_j$ is the position of first word in
    question and $end_j$ is the position of last word in question
Output:
    A set of retokenized subsequence $\{TSub_1, TSub_2, .....\}$
    $N_t$: the number of retokenized subsequence
Algorithm:
    Initially, we set $N_t$ := 1; $TSub_1$:=$QW_{start_j}$;
    if ($Sub_j^Q \neq \psi$)
    {   /*** from the start to the end positions in the string ***/
        for ( $k$ := $start_j$+1 to $end_j$)
        {
  /***Check the two question words is bigram in the passage***/
        if (bigram($QW_{k-1}$,$QW_k$) is_found_in_passage)
            add $QW_k$ into $TSub_{N_t}$;
        Otherwise
        {   $N_t$ ++;
            $TSub_{N_t}$ := $QW_k$;
        } /*** End otherwise***/
        } /*** End for ***/
    } /*** End if ***/
    else
        $N_t$ := 0;
```

Figure 3: An algorithm for retokenizing subsequence

```
Algorithm 2: Copmuting_DP_score
Input:
    A subsequence  $Sub_j^Q$ where $start_j$ is the position of first word
    of $Sub_j^Q$ in question $end_j$ is the position of last word of $Sub_j^Q$ in
    question
Output:
    The score of DP($Sub_j^Q$)
Algorithm:
    head := $start_j$;
    tail := $end_j$;
    Max_score := 0;
    for ($k$ := head ~ tail)
    {   let $WORD$ := $QW_k$, $QW_{k+1}$,…, $QW_{tail}$;
        /*** look-up $WORD$ in the index files  ***/
        compute DP($WORD$) using equation (3);
        if (DP($WORD$) > Max_score)
            Max_score := DP($WORD$);
    } /*** End for ***/
    DP($WORD$) := Max_score;
```

Figure 4: An algorithm for computing DP score for a subsequence

The density scoring can be thought as measuring "how much information the passage preserves in response to the question". On the contrary, the QW_Weight (second term in equation (1)) aims to estimate "how much content information the passage has given the question". To achieve this, we further take the other extracted common subsequences, i.e., Sub_j^{P} into account. By means of the same term weighting schema for the set of Sub_j^{P}, the QW_Weight is then produced. Equation (8) gives the overall QW_Weight measurement.

$$\text{QW\_Weight}(\text{Q},\text{P}) = \sum_{i=1}^{S\_CNT} \text{Weight}(\text{Sub}_i^{\text{P}}) = \sum_{i=1}^{S\_CNT} (\text{length}(\text{Sub}_i^{\text{P}})^{\alpha_1} \times \text{DP}(\text{Sub}_i^{\text{P}})) \tag{8}$$

where the DP score of the input subsequence can be obtained via the algorithm 2 (Figure 5). *S_CNT* is the number of subsequence in P. The parameter α_1 is also set as equal as equation (2).

In addition, the neighboring contexts of a sentence, which contains high QW_Density score might include the answers. Hence, we stress on either head or tail fragments of the passage. In other words, the passage score is determined by computing equation (1) for head and tail parts of passage. We thus extend equation (1) as follows.

$$\begin{aligned}\text{Passage\_Score}(\text{P}) = \max\{ & \lambda \times \text{QW\_Density}(\text{Q},\text{P}_1) + (1-\lambda) \times \text{QW\_Weight}(\text{Q},\text{P}_1),\\ & \lambda \times \text{QW\_Density}(\text{Q},\text{P}_2) + (1-\lambda) \times \text{QW\_Weight}(\text{Q},\text{P}_2)\}\end{aligned}$$

$$\begin{cases} \text{if P has 3 sentences}: & \text{S}_1, \text{S}_2, \text{S}_3 & \text{then, } \text{P}_1 = \text{S}_1 + \text{S}_2 \text{ and } \text{P}_2 = \text{S}_2 + \text{S}_3 \\ \text{else if P has 2 sentences}: & \text{S}_1, \text{S}_2 & \text{then, } \text{P}_1 = \text{S}_1 \text{ and } \text{P}_2 = \text{S}_2 \\ \text{else if P has 1 sentence}: & \text{S}_1 & \text{then, } \text{P}_1 = \text{P}_2 = \text{S}_1 \end{cases}$$

Instead of estimating the whole passage, the two divided parts: P_1, and P_2 are used. We select the maximum passage score from either head (P_1) or tail (P_2) part. When the passage contains only one sentence, then this sentence is indispensable to be used for estimation.

Now we turn to analyze the time complexity of our algorithm. It is known that the suffix tree construction costs is linear time (assume it requires $O(T)$, T: the passage length for passage and $O(T')$, T': the question length for question). Assume the search time for a pattern in the suffix trees is at most $O(h\log m)$ where h is the tree height, and m is the number of branch nodes. To generate the sets of Sub_j^{Q} and Sub_j^{P}, it involves in building suffix trees and incrementally searching substrings, i.e., $O((T+T')+(T+T')(h\log m))$. Intuitively, both algorithm 1, and algorithm 2 are linear time algorithms, which depends on the length of "common" subsequence, i.e., at most $O(\min(T, T'))$. Consequently, the overall time complexity of our method for computing a passage is $O((T+T')(1+h\log m)+\min(T, T'))$.

4 Experiments

4.1 Evaluation

We should carefully select the use of videoQA collection for evaluation. Unfortunately, there is no benchmark corpus for this task. Thus, we develop an annotated collection by following the similar tasks as TREC, CLEF, and NTCIR. The Discovery videos are one of the popular raw video sources and widely evaluated in many literatures (Lin et al., 2001; Wu et al., 2004; Lee et al., 2005). Totally, 75.6 hours of Discovery videos (93 video names) were used. Table 1 lists the statistics of the Discovery films.

The questions were created in two different ways: one set (about 73) was collected from previous studies (Lin et al., 2001; Wu et al., 2004) which came from the "Project: Assignment of Discovery"; while the other was derived from a real log from users. Video collections are difficult to be general-purpose since hundreds hours of videos might take tens of hundreds GB storage space. Therefore, general questions are quite difficult to be found in the video database. Hence, we provide a list of short introductions collected from the cover-page of the videos and enable users to browse the descriptions. Users were then asked for the system with limited to the collected video topics. We finally filter the (1) keyword-like queries (2) non-Chinese and (3) un-supported questions. Finally, there were 253 questions for evaluation.

For the answer assessment, we followed the TREC-QA track (Voorhees, 2001) and NTCIR to annotate answers in the pool that collected from the outputs of different passage retrieval methods. Unlike traditional text QA task, most of the OCR sentences contain a number of OCR error words. Furthermore, some sentence did include the answer string but error recognized as different words. Thus, instead of annotating the recognized transcripts, we used the corresponding video frames for evaluation because users can directly find the answers in the retrieved video clips and recognized text. Among 253 questions, 56 of which did not have an answer, while 368 passage&frame segments (i.e., answer patterns) in the pool were labeled as answers. On averagely, there are 1.45 labeled answers for each question.

The MRR (Voorhees, 2001) score, precision and pattern-recall are used for evaluation. We measure the MRR scores for both top1 and top5 ranks, and precision and pattern-recall rates for top5 retrieved answers.

Table 1: Statistics of the collected Discovery videos

# of videos	# of sentence	# of words	# of passages
93	49950	746276	25001
AVG # of words per sentence	AVG # of words per passage	AVG # of sentences per passage	AVG # of words per video
14.94	48.78	537.09	8024.47

4.2 Results

In this paper, we employed six top-performed yet portable ranking models, TFIDF, BM-25 (Robertson et al., 2000), INQUERY, language model (Zhai and Lafferty, 2001), cosine, and density-based (Lee et al., 2001) approaches for comparison[1]. For the language model, the Jelinek-Mercer smoothing method was employed with the parameter settings λ=0.5 which was selected via several trials. In our preliminary experiments, we found that the query term expansion does not improve but decrease the overall ranking performance for all the ranking models. Thus, we only compare with the "pure" retrieval performance without pseudo-feedback.

The system performance was evaluated through the returned passages. We set α_1=1.25, α_2= 0.25, and λ=0.8 which were observed via the following parameter validations. More detail parameter experiments are presented and discussed later. Table 2 lists the overall videoQA results with different ranking models.

Among all ranking models, the proposed method achieves the best system performance. Our approach produced 0.596 and 0.654 MRR scores when evaluating the top1 and top5 passages and the precision rate achieves 0.208. Compared to the second best method (language model), our method is 10.16% better in relatively percentage in terms of MRR(top1) score. For the MRR(top5) score, our method is 7.39 relative percentage better. In terms of the non-answered questions, our method also covers the most questions (253-69=184) compared to the other ranking models. Overall, the experiment shows that the proposed weighted string pattern matching algorithm outperforms the other six methods in terms of MRR, non-answered question numbers, precision and pattern recall rates.

[1] For the TFIDF/BM-25/INQUERY/Language Model approaches were performed using the Lemur toolkit

Table 2: Overall videoQA performance with different ranking models (using unigram Chinese word)

Word-Level	MRR (Top1)	MRR (Top5)	Non-answered Questions	Precision	Pattern Recall
TFIDF	0.498	0.572	81	0.189	0.649
BM-25	0.501	0.581	78	0.186	0.638
Language Model	0.541	0.609	74	0.196	0.671
INQUERY	0.505	0.583	78	0.188	0.644
Cosine	0.418	0.489	102	0.151	0.519
Density	0.323	0.421	102	0.137	0.471
Our Method	**0.596**	**0.654**	**69**	**0.208**	**0.711**

Table 3: Overall videoQA performance with different ranking models using word segmentation tools

Word-Level	MRR (Top1)	MRR (Top5)	Non-answered Questions	Precision	Pattern Recall
TFIDF	**0.509**	**0.567**	**89**	0.145	**0.597**
BM-25	0.438	0.500	104	0.159	0.543
Language Model	0.486	0.551	89	0.172	0.589
INQUERY	0.430	0.503	97	0.164	0.562
Cosine	0.403	0.480	100	0.158	0.548
Density	0.304	0.380	125	0.133	0.451
Our Method	**0.509**	**0.561**	**89**	**0.181**	**0.608**

Next, we evaluate the performance with adopting a trained Chinese word segmentation tool instead of unigram level of word. In this paper, we employed the Chinese word segmentation tool (Wu et al., 2006) that achieved about 0.93-0.96 recall/precision rates in the SIGHAN-3 word segmentation task (Levow, 2006). Table 3 lists the overall experimental results with the adopted word segmentation tool. In comparison to unigram grained level (Table 2), it is shown that the use of word segmentation tool does not improve the videoQA result for most top-performed ranking models, BM-25, language model, INQUERY, and our method. For example, our method is relatively 17.92% and 16.57% worse in MRR(Top1) and MRR(Top5) scores. In terms of precision and pattern-recall rates, it drops 14.91, and 16.94 relative percentages, respectively. For the TFIDF method, the MRR score is almost the same as previous result whereas it decreased 30.34%, and 8.71% precision and pattern-recall rates. On averagely, the four models, BM-25, language model, INQUERY, and our method dropped at least relatively 10% in MRR, precision, and pattern-recall rates. In this experiment, our ranking algorithm also achieved

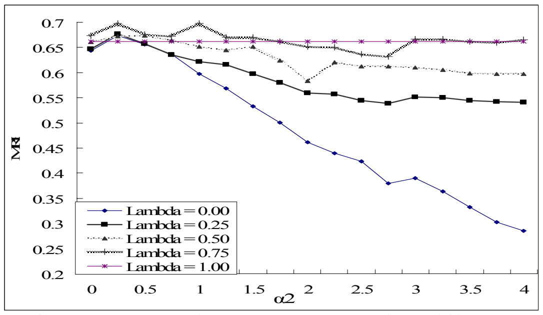


Figure 5: Experimental results with different settings of parameter α_1 using MRR evaluation

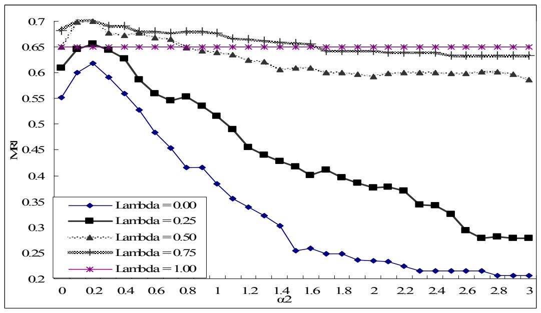


Figure 6: Verify parameter α_2 with α_1=1.25, and variant λ

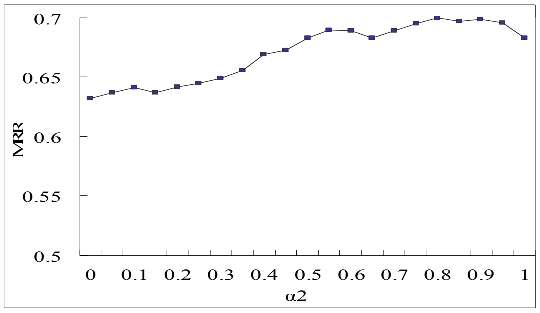


Figure 7: Verify parameter λ in the two validation sets with α_1=1.25 and α_2=0.25

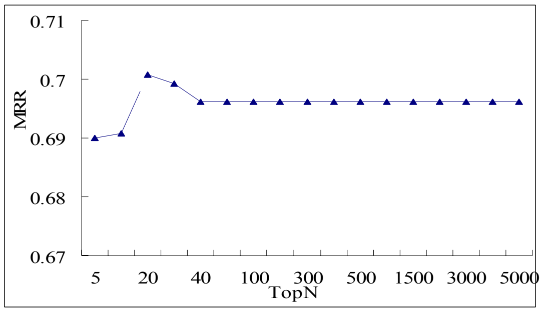


Figure 8: Experimental results with different number of initial retrieval passages (Top*N*)

the best results in terms of precision and pattern recall rates while marginally worse than the TFIDF for the MRR(top5) score.

There are three parameters: λ, α_1, α_2, in our ranking algorithm. λ controls the weight of the QW_Density(Q, P), while α_1, and α_2 were set for the power of subsequence length and the distance measurement. We randomly select 100 questions for parameter validations. Firstly, we tried to verify the optimal α_1 via different settings of the remaining two parameters. The best α_1 is then set to verify α_2 via various λ values. The optimal λ is subsequently confirmed through the observed α_1 and α_2 values. Figure 5, 6, 7 show the performance evaluations of different settings for the three parameters.

As shown in Figure 5, the optimal settings of (α_1=1.25) is obtained when and α_2=0.25, and λ=0.75. When α_1 is set more than 1.5, our method quickly decreased. In this experiment, we also found that large α_2 negatively affects the performance. The small α_2 values often lead to better ranking performance. Thus, in the next experiment, we limited the α_2 value in 0.0~3.0. As seeing in Figure 6, again the abnormal high or zero α_2 values give the poor results. This implies the over-weight and no-weight on the distance measurement (equation (7)) is not useful. Instead, a small α_2 value yields to improve the performance. In our experiment, α_2=0.25 is quite effective. Finally, in Figure 7, we can see that both taking the QW_Density, and QW_Weight into account gives better ranking result, especially QW_Density. This experiment indicates that the combination of QW_Density and QW_Weight is better than its individual term weighting strategy. When λ=0.8, the best ranking result (MRR = 0.700) is reached.

Next, we address on the impact of different number of initial retrieved passages using BM-25 ranking models. Due to the length limitation of this paper, we did not present the experiments over all the compared ranking models, while we left the further results at our web site[2]. For the three parameters, we select the optimal settings derived from previous experimental results, i.e., λ=0.8, α_1=1.25, α_2=0.25. Figure 8 shows the experimental results with different number of initial retrieved passages. When employing exactly five initial retrieved passages, it can be viewed as the re-ranking improvement over the BM-25 ranking model. As seeing in Figure 8, our method does improve the conventional BM-25 ranking approach (MRR score 0.690 v.s. 0.627) with relatively 10.04% MRR value. The best system performance is MRR=0.700 when there are merely 20 initial retrieved passages. The ranking result converges when retrieving more than 40 passages. Besides,

[2] http://140.115.112.118/bcbb/TVQS2/

we also continue the experiments using only top-20 retrieved passages on the actual 253 testing questions. The ranking performance is then further enhanced from MRR=0.654 to 0.663 with 1.37% relatively improved.

5 Conclusion

More and more users are interested in searching for answers in videos, while existing question answering systems do not support multimedia accessing. This paper presents a weighted string pattern matching-based passage ranking algorithm for extending text QA toward video question answering. We compare our method with six top-performed ranking models and show that our method outperforms the second best approach (language model) in relatively 10.16 % MRR score, and 6.12% precision rates.

In the future, we plan to integrate the other useful features in videos to support multi-model-based multimedia question answering. The video-demo version of our videoQA system can be found at the web site (http://140.115.112.118/bcbb/TVQS2/).

References

Cao, J., and Nunamaker J. F. Question answering on lecture videos: a multifaceted approach, International Conference on Digital Libraries, pages 214 – 215, 2004.

Chang, F., Chen, G. C., Lin, C. C., and Lin, W. H. Caption analysis and recognition for building video indexing systems. Multimedia systems, 10: 344-355, 2005.

Cui, H., Sun, R., Li, K., Kan, M., and Chua, T. Question answering passage retrieval using dependency relations. In Proceedings of the 28th ACM SIGIR Conference on Research and Development in Information Retrieval, pages 400-407, 2005.

Fan, J., Yau, D. K. Y., Elmagarmid, A. K., and Aref, W. G. Automatic image segmentation by integrating color-edge extraction and seeded region growing. IEEE Trans. On Image Processing, 10(10): 1454-1464, 2001.

Fung, P., Ngai, G., Yuan, Y., and Chen, B. A maximum entropy Chinese parser augmented by transformation-based learning. ACM Trans. Asian Language Information Processing, 3: 159-168, 2004.

Lee et al. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In Proceedings of the 10th Text Retrieval Conference, pages 437-446, 2001.

Lee, Y. S., Wu, Y. C., and Chang, C. H. Integrating Web information to generate Chinese video summaries. In Proceedings of 17th international conference on software engineering and knowledge engineering (SEKE), pages 514-519, 2005.

Levow, G. A. The third international Chinese language processing Bakeoff: word segmentation and named entity recognition, In Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing, pages 108-117, 2006.

Lin, C. J., Liu, C. C., and Chen, H. H. A simple method for Chinese videoOCR and its application to question answering. Journal of Computational linguistics and Chinese language processing, 6: 11-30, 2001.

Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., and Karger, D. R. What makes a good answer? the role of context in question answering. In Proceedings of the 9th international conference on human-computer interaction (INTERACT), pages 25-32, 2003.

Lienhart, R. and Wernicke, A. Localizing and segmenting text in images and videos. IEEE Trans. Circuits and Systems for Video Technology, 12(4): 243-255, 2002.

Lyu, M. R., Song, J., and Cai, M. A comprehensive method for multilingual video text detection, localization, and extraction. IEEE Trans. Circuits and Systems for Video Technology, 15(2): 243-255, 2005.

Over, P., Ianeva, T., Kraaij, W., and Smeaton, A. F. TRECVID 2005 - an overview. In Proceedings of the 14th text retrieval conference (TREC), 2005.

Pasca, M., and Harabagiu, S. High-performance question answering. In Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval, pages 366-374, 2001.

Robertson, E., Walker, S., and Beaulieu, M. Experimentation as a way of life: Okapi at TREC. Journal of Information processing and management, 36: 95-108, 2000.

Tellex, S., Katz, B., Lin, J. J., Fernandes, A., and Marton, G. Quantitative evaluation of passage retrieval algorithms for question answering. In Proceedings of the 26th ACM SIGIR Conference on Research and Development in Information Retrieval, pages 41-47, 2003.

Voorhees, E. M. Overview of the TREC 2001 question answering track. In Proceedings of the 10th Text Retrieval Conference , pages 42-52, 2001.

Ukkonen, E. Constructing suffix trees on-line in linear time. In Proceedings of the international federation of information processing, pages 484-492, 1995.

Wactlar, H. D. Informedia search and summarization in the video medium, In Proceedings of Imagina 2000 Conference, 2000.

Wu, Y. C., Lee, Y. S., Chang, C. H. CLVQ: Cross-language video question/answering system. In Proceedings of 6th IEEE International Symposium on Multimedia Software Engineering, pages 294-301, 2004.

Wu, Y. C., Yang, J. C., and Lin, Q. X. Description of the NCU Chinese Word Segmentation and Named Entity Recognition System for SIGHAN Bakeoff 2006. In Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing, pages 209-212, 2006.

Yang, H., Chaison, L., Zhao, Y., Neo, S. Y., and Chua, T. S. VideoQA: Question answering on news video. In Proceedings of the 11th ACM International Conference on Multimedia, pages 632-641, 2003.

Zhai, C., and Lafferty, J. 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 (SIGIR), pages 334-342, 2001.

Zhang, D., and Nunamaker, J. A natural language approach to content-based video indexing and retrieval for interactive E-learning. IEEE Trans. on Multimedia, 6: 450-458, 2004.

Can Semantic Roles Generalize Across Genres?

Szu-ting Yi
Dept of Computer Science
University of Pennsylvania
Philadelphia, PA 19104

Edward Loper
Dept of Computer Science
University of Pennsylvania
Philadelphia, PA 19104

Martha Palmer
Dept of Computer Science
University of Colorado at Boulder
Boulder, CO 80309

Abstract

PropBank has been widely used as training data for Semantic Role Labeling. However, because this training data is taken from the WSJ, the resulting machine learning models tend to overfit on idiosyncrasies of that text's style, and do not port well to other genres. In addition, since PropBank was designed on a verb-by-verb basis, the argument labels Arg2 - Arg5 get used for very diverse argument roles with inconsistent training instances. For example, the verb "make" uses Arg2 for the "Material" argument; but the verb "multiply" uses Arg2 for the "Extent" argument. As a result, it can be difficult for automatic classifiers to learn to distinguish arguments Arg2-Arg5. We have created a mapping between PropBank and VerbNet that provides a VerbNet thematic role label for each verb-specific PropBank label. Since VerbNet uses argument labels that are more consistent across verbs, we are able to demonstrate that these new labels are easier to learn.

1 Introduction

Correctly identifying semantic entities and successfully disambiguating the relations between them and their predicates is an important and necessary step for successful natural language processing applications, such as text summarization, question answering, and machine translation. For example, in order to determine that question (1a) is answered by sentence (1b), but not by sentence (1c), we must determine the relationships between the relevant verbs (*eat* and *feed*) and their arguments.

(1) a. What do lobsters like to eat?
b. Recent studies have shown that lobsters primarily feed on live fish, dig for clams, sea urchins, and feed on algae and eel-grass.
c. In the early 20th century, Mainers would only eat lobsters because the fish they caught was too valuable to eat themselves.

An important part of this task is *Semantic Role Labeling* (SRL), where the goal is to locate the constituents which are arguments of a given verb, and to assign them appropriate semantic roles that describe how they relate to the verb. Many researchers have investigated applying machine learning to corpus specifically annotated with this task in mind, PropBank, since 2000 (Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Hacioglu et al., 2003; Moschitti, 2004; Yi and Palmer, 2004; Pradhan et al., 2005b; Punyakanok et al., 2005; Toutanova et al., 2005). For two years, the CoNLL workshop has made this problem the shared task (Carreras and Márquez, 2005). However, there is still little consensus in the linguistic and NLP communities about what set of role labels are most appropriate. The Proposition Bank (PropBank) corpus (Palmer et al., 2005) avoids this issue by using theory-agnostic labels (Arg0, Arg1, ..., Arg5), and by defining those labels to have verb-specific meanings. Under this scheme, PropBank can avoid making any claims

Proceedings of NAACL HLT 2007, pages 548–555,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

about how any one verb's arguments relate to other verbs' arguments, or about general distinctions between verb arguments and adjuncts.

However, there are several limitations to this approach. The first is that it can be difficult to make inferences and generalizations based on role labels that are only meaningful with respect to a single verb. Since each role label is verb-specific, we can not confidently determine when two different verbs' arguments have the same role; and since no encoded meaning is associated with each tag, we can not make generalizations across verb classes. In contrast, the use of a shared set of role labels, such as thematic roles, would facilitate both inferencing and generalization.

The second issue with PropBank's verb-specific approach is that it can make training automatic semantic role labeling (SRL) systems more difficult. A vast amount of data would be needed to train the verb-specific models that are theoretically mandated by PropBank's design. Instead, researchers typically build a single model for the numbered arguments (Arg0, Arg1, . . . , Arg5). This approach works surprisingly well, mainly because an explicit effort was made to use arguments Arg0 and Arg1 consistently across different verbs; and because those two argument labels account for 85% of all arguments. However, this approach causes the system to conflate different argument types, especially with the highly overloaded arguments Arg2-Arg5. As a result, these argument labels are quite difficult to learn.

A final difficulty with PropBank's current approach is that it limits SRL system robustness in the face of verb senses, verbs or verb constructions that were not included in the training data, and the training data is all Wall Street Journal corpora. If a PropBank-trained SRL system encounters a novel verb or verb usage, then there is no way for it to know which role labels are used for which argument types, since role labels are defined so specifically. This is especially problematic for Arg2-5. Similarly, PropBank-trained SRL systems can have difficulty generalizing when a known verb is encountered in a novel construction. These problems can happen quite frequently if the training data comes from a different genre than the test data. This issue is reflected in the relatively poor performance of most state-of-the-art SRL systems when tested on a novel genre, the Brown corpus, during CoNLL 2005. For example, the SRL system described in (Pradhan et al., 2005b; Pradhan et al., 2005a) achieves an F-score of 81% when tested on the same genre as it is trained on (WSJ); but that score drops to 68.5% when the same system is tested on a different genre (the Brown corpus). DARPA-GALE is funding an ongoing effort to PropBank additional genres, but better techniques for generalizing the semantic role labeling task are still needed.

In this paper, we demonstrate an increase in the generality of our semantic role labeling based on a mapping that has been developed between PropBank and another lexical resource, VerbNet. By taking advantage of VerbNet's more consistent set of labels, we can generate more useful role label annotations with a resulting improvement in SRL performance on novel genres.

2 Background

2.1 PropBank

PropBank (Palmer et al., 2005) is an annotation of one million words of the Wall Street Journal portion of the Penn Treebank II (Marcus et al., 1994) with predicate-argument structures for verbs, using semantic role labels for each verb argument. In order to remain theory neutral, and to increase annotation speed, role labels were defined on a per-verb-sense basis. Although the same tags were used for all verbs, (namely Arg0, Arg1, ..., Arg5), these tags are meant to have a verb-specific meaning.

Thus, the use of a given argument label should be consistent across different uses of that verb, including syntactic alternations. For example, the Arg1 (underlined) in "John broke the window" is the same window that is annotated as the Arg1 in "The window broke", even though it is the syntactic subject in one sentence and the syntactic object in the other. However, there is no guarantee that an argument label will be used consistently across different verbs. For example, the Arg2 label is used to designate the *destination* of the verb "bring;" but the *extent* of the verb "rise." Generally, the arguments are simply listed in the order of their prominence for each verb. However, an explicit effort was made when PropBank was created to use Arg0 for arguments that fulfill Dowty's criteria for "prototypical agent," and Arg1 for arguments that fulfill the criteria for "prototypical patient." (Dowty, 1991) As a result, these two argument labels are significantly more consistent across verbs than the other three. But nevertheless, there are still some inter-verb inconsistencies for even Arg0 and Arg1.

2.2 VerbNet

VerbNet (Schuler, 2005) consists of hierarchically arranged verb classes, inspired by and extended from classes of Levin 1993 (Levin, 1993). Each class and subclass is characterized extensionally by its set of verbs, and intensionally by a list of the arguments of those verbs and syntactic and semantic information about the verbs. The argument list consists of thematic roles (23 in total) and possible selectional restrictions on the arguments expressed using binary predicates. The syntactic information maps the list of thematic arguments to deep-syntactic arguments (i.e., normalized for voice alternations, and transformations). The semantic predicates describe the participants during various stages of the event described by the syntactic frame.

The same thematic role can occur in different classes, where it will appear in different predicates, providing a class-specific interpretation of the role. VerbNet has been extended from the original Levin classes, and now covers 4526 senses for 3769 verbs. A primary emphasis for VerbNet is the grouping of verbs into classes that have a coherent syntactic and semantic characterization, that will eventually facilitate the acquisition of new class members based on observable syntactic and semantic behavior. The hierarchical structure and small number of thematic roles is aimed at supporting generalizations.

2.3 Mapping PropBank to VerbNet

Because PropBank includes a large corpus of manually annotated predicate-argument data, it can be used to train supervised machine learning algorithms, which can in turn provide PropBank-style annotations for novel or unseen text. However, as we discussed in the introduction, PropBank's verb-specific role labels are somewhat problematic. Furthermore, PropBank lacks much of the information that is contained in VerbNet, including information about selectional restrictions, verb semantics, and inter-verb relationships.

We have therefore created a mapping between VerbNet and PropBank (Loper et al., 2007), which will allow us to use the machine learning techniques that have been developed for PropBank annotations to generate more semantically abstract VerbNet representations. Additionally, the mapping can be used to translate PropBank-style numbered arguments (Arg0…Arg5) to VerbNet thematic roles (Agent, Patient, Theme, etc.), which should allow us to overcome the verb-specific nature of PropBank.

The mapping between VerbNet and PropBank consists of two parts: a *lexical mapping* and an *instance classifier*. The lexical mapping is responsible for specifying the potential mappings between PropBank and VerbNet for a given word; but it does not specify which of those mappings should be used for any given occurrence of the word. That is the job of the instance classifier, which looks at the word in context, and decides which of the mappings is most appropriate. In essence, the instance classifier is performing word sense disambiguation, deciding which lexeme from each database is correct for a given occurrence of a word. In order to train the instance classifier, we semi-automatically annotated each verb in the PropBank corpus with VerbNet class information.[1] This *mapped corpus* was then used to build the instance classifier. More details about the mapping, and how it was created, can be found in (Loper et al., 2007).

3 Analysis of the Mapping

In order to confirm our belief that PropBank roles Arg0 and Arg1 are relatively coherent, while roles Arg2-5 are much more overloaded, we performed a preliminary analysis of how argument roles were mapped. Figure 1 shows how often each PropBank role was mapped to each VerbNet thematic role, calculated as a fraction of instances in the mapped corpus. From this figure, we can see that Arg0 maps to agent-like roles, such as "agent" and "experiencer," over 94% of the time; and Arg1 maps to patient-like roles, including "theme," "topic," and "patient," over 82% of the time. In contrast, arguments Arg2-5 get mapped to a much broader variety of roles. It is also worth noting that the sample size for arguments

[1]Excepting verbs whose senses are not present in VerbNet (24.5% of instances).

Arg3-5 is quite small in comparison with arguments Arg0-2, suggesting that any automatically built classifier for arguments Arg3-5 will suffer severe sparse data problems for those arguments.

4 Training a SRL system with VerbNet Roles to Achieve Robustness

An important issue for state-of-the-art automatic SRL systems is robustness: although they receive high performance scores when tested on the Wall Street Journal (WSJ) corpus, that performance drops significantly when the same systems are tested on a corpus from another genre. This performance drop reflects the fact that the WSJ corpus is highly specialized, and tends to use genre-specific word senses for many verbs. The 2005 CoNLL shared task has addressed this issue of robustness by evaluating participating systems on a test set extracted from the Brown corpus, which is very different from the WSJ corpus that was used for training. The results suggest that there is much work to be done in order to improve system robustness.

One of the reasons that current SRL systems have difficulty deciding which role label to assign to a given argument is that role labels are defined on a per-verb basis. This is less problematic for Arg0 and Arg1, where a conscious effort was made to be consistent across verbs; but is a significant problem for Args[2-5], which tend to have very verb-specific meanings. This problem is exacerbated even further on novel genres, where SRL systems are more likely to encounter unseen verbs and uses of arguments that were not encountered in the training data.

4.1 Addressing Current SRL Problems via Lexical Mappings

By exploiting the mapping between PropBank and VerbNet, we can transform the data to make it more consistent, and to expand the size and variety of the training data. In particular, we can use the mapping to transform the verb-specific PropBank role labels into the more general thematic role labels that are used by VerbNet. Unlike the PropBank labels, the VerbNet labels are defined consistently across verbs; and therefore it should be easier for statistical SRL systems to model them. Furthermore, since the VerbNet role labels are significantly less verb-

Arg0 (45,579)	
Agent	85.4%
Experiencer	7.2%
Theme	2.1%
Cause	1.9%
Actor1	1.8%
Theme1	0.8%
Patient1	0.2%
Location	0.2%
Theme2	0.2%
Product	0.1%
Patient	0.0%
Attribute	0.0%

Arg1 (59,884)	
Theme	47.0%
Topic	23.0%
Patient	10.8%
Product	2.9%
Predicate	2.5%
Patient1	2.4%
Stimulus	2.0%
Experiencer	1.9%
Cause	1.8%
Destination	0.9%
Theme2	0.7%
Location	0.7%
Source	0.7%
Theme1	0.6%
Actor2	0.6%
Recipient	0.5%
Agent	0.4%
Attribute	0.2%
Asset	0.2%
Patient2	0.2%
Material	0.2%
Beneficiary	0.0%

Arg2 (11,077)	
Recipient	22.3%
Extent	14.7%
Predicate	13.4%
Destination	8.6%
Attribute	7.6%
Location	6.5%
Theme	5.5%
Patient2	5.3%
Source	5.2%
Topic	3.1%
Theme2	2.5%
Product	1.5%
Cause	1.2%
Material	0.8%
Instrument	0.6%
Beneficiary	0.5%
Experiencer	0.3%
Actor2	0.2%
Asset	0.0%
Theme1	0.0%

Arg3 (609)	
Asset	38.6%
Source	25.1%
Beneficiary	10.7%
Cause	9.7%
Predicate	9.0%
Location	2.0%
Material	1.8%
Theme1	1.6%
Theme	0.8%
Destination	0.3%
Instrument	0.3%

Arg4 (18)	
Beneficiary	61.1%
Product	33.3%
Location	5.6%

Arg5 (17)	
Location	100.0%

Figure 1: The frequency with which each PropBank numbered argument is mapped to each VerbNet thematic role in the mapped corpus. The numbers next to each PropBank argument reflects the number of occurrences of that numbered argument in the mapped corpus.

dependent than the PropBank roles, the SRL's models should generalize better to novel verbs, and to novel uses of known verbs.

5 SRL Experiments on Linked Lexical Resources

In order to verify the feasibility of performing semantic role labeling with VerbNet thematic roles, we re-trained our existing SRL system, which originally used PropBank role labels, with a new label set that makes use of VerbNet thematic role information.

5.1 The SRL System

Our SRL system is a Maximum Entropy based pipelined system which consists of four components: Pre-processing, Argument Identification, Argument Classification, and Post Processing. The Pre-processing component pipes a sentence through a syntactic parser and filters out constituents which are unlikely to be semantic arguments based on a constituents location in the parse tree. The Argument Identification component is a binary MaxEnt classifier, which tags candidate constituents as arguments or non-arguments. The Argument Classification component is a multi-class MaxEnt classifier which assigns a semantic role to each constituent. The Post Processing component further selects the final arguments based on global constraints. Our experiments mainly focused on changes to the Argument Classification stage of the SRL pipeline, and in particular, on changes to the set of output tags. For more information on our SRL system, see (Yi and Palmer, 2004; Yi and Palmer, 2005).

The evaluation of SRL systems is typically expressed by precision, recall and the F1-measure. Precision is the number of correct arguments predicted by a system divided by the total number of arguments proposed. Recall is the number of correct arguments divided by the number of the total number of arguments in the Gold Standard Data. F1 computes the harmonic mean of precision and recall.

5.2 SRL Experiments on Mapped VerbNet Thematic Roles

Since PropBank arguments Arg0 and Arg1 are already quite coherent, we left them as-is in the new label set. But since arguments Arg2-Arg5 are highly

Group 1	Group 2	Group 3	Group 4	Group 5
Recipient	Extent	Predicate	Patient2	Instrument
Destination	Asset	Attribute	Product	Cause
Location		Theme		Experiencer
Source		Theme1		Actor2
Material		Theme2		
Beneficiary		Topic		

Figure 2: Thematic Role Groupings for the experiments on linked lexical resources; and for Arg2 in the experiments on arguments with different verb independency.

overloaded, we replaced them by mapping them to their corresponding VerbNet thematic role. We found that mapping directly to individual role labels created a significant sparse data problem, since the number of output tags was increased from 6 to 23. We therefore grouped the VerbNet thematic roles into five coherent groups of similar thematic roles, shown in Figure 2.[2] Our new tag set therefore included the following tags: **Arg0** (*agent*); **Arg1** (*patient*); **Group1** (*goal*); **Group2** (*extent*); **Group3** (*predicate/attrib*); **Group4** (*product*); and **Group5** (*instrument/cause*).

Training our SRL system using these thematic role groups, we obtained performance similar to the original SRL system. However, it is important to note that these performance figures are not directly comparable, since the two systems are performing different tasks: The Original system labels Arg0-5,ArgA and ArgM and the Mapped system labels Arg0, Arg1, ArgA, ArgM and Group1-5. In particular, the role labels generated by the original system are verb-specific, while the role labels generated by the new system are less verb-dependent.

5.2.1 Results

For our testing and training, we used the portion of Penn Treebank II that is covered by the mapping, and where at least one of Arg2-5 is used. Training was performed using sections 2-21 of the Treebank (10,783 instances of argument); and testing was performed on section 23 (859 instances). Table 1 displays the performance score for the SRL system using the augmented tag set ("Mapped"). The performance score of the original system ("Original") is also listed, for reference; however, as was dis-

[2] Karin Kipper assisted in creating the groupings.

System	Precision	Recall	F1
Original	90.65	85.43	87.97
Mapped	88.85	84.56	86.65

Table 1: Overall SRL System performance using the PropBank tag set ("Original") and the augmented tag set ("Mapped")

System	Precision	Recall	F1
Original	97.60	83.67	90.10
Mapped	91.70	82.86	87.06

Table 2: SRL System performance evaluated on only Arg2-5 (Original) or Group1-5 (Mapped).

cussed above, these results are not directly comparable because the two systems are performing different tasks.

The results indicate that the performance drops when we train on the new argument labels, especially on precision when we evaluate the systems on only Arg2-5/Group1-5 (see Table 2). However, it is premature to conclude that there is no benefit from the VerbNet thematic role labels. Firstly, we have very few mapped Arg3-5 instances (less than 1,000 instances); secondly, we lack test data generated from a genre other than WSJ to allow us to evaluate the robustness (generality) of SRL trained on the new argument labels.

We therefore redesigned our experiments by limiting the scope to mapped instances of Arg1 and Arg2. By doing this, we should be able to accomplish the following: 1) we can map new argument labels back to the original PropBank labels; therefore we can directly compare results; 2) With the ability of testing our systems on other test data, we can evaluate the influence of the mapping on SRL robustness; 3) We can validate our original hypothesis that the behavior of Arg1 is primarily verb-independent while Arg2 is more verb-specific.

5.3 SRL Experiments on Arguments with Different Verb Independency

We conducted two further sets of experiments: one to test the effect of the mapping on learning Arg2; and one to test the effect on learning Arg1. Since Arg2 is used in very verb-dependent ways, we expect that mapping it to VerbNet role labels will in-

Group 1	Group 2	Group 3	Group 4	Group 5
Theme	Source	Patient	Agent	Topic
Theme1	Location	Product	Actor2	
Theme2	Destination	Patient1	Experiencer	Group 6
Predicate	Recipient	Patient2	Cause	Asset
Stimulus	Beneficiary			
Attribute	Material			

Figure 3: Thematic Role Groupings for Arg1 in the experiments on arguments with different verb independency.

crease our performance. However, since a conscious effort was made to keep the meaning of Arg1 consistent across verbs, we expect that mapping it to VerbNet labels will provide less of an improvement.

Each experiment compares two SRL systems: one trained using the original PropBank role labels; the other trained with the argument role under consideration (Arg1 or Arg2) subdivided based on which VerbNet role label it maps to. In order to prevent the training data from these subdivided labels from becoming too sparse (which would impair system performance) we grouped similar thematic roles together. For Arg2, we used the same groupings as the previous experiment, shown in Figure 2. The argument role groupings we used for Arg1 are shown in Figure 3.

The training data for both experiments is the portion of Penn Treebank II (sections 02-21) that is covered by the mapping. We evaluated each experimental system using two test sets: section 23 of the Penn Treebank II, which represents the same genre as the training data; and the PropBank-ed portion of the Brown corpus, which represents a very different genre.

5.3.1 Results and Discussion

Table 3 describes the results of SRL overall performance tested on the WSJ corpus Section 23; Table 4 demonstrates the SRL overall system performance tested on the Brown corpus. Systems Arg1-Original and Arg2-Original are trained using the original PropBank labels, and show the baseline performance of our SRL system. Systems Arg1-Mapped and Arg2-Mapped are trained using PropBank labels augmented with VerbNet thematic role groups. In order to allow comparison between the system using the original PropBank labels and the systems that augmented those labels with VerbNet

System	Precision	Recall	F1
Arg1-Original	89.24	77.32	82.85
Arg1-Mapped	90.00	76.35	82.61
Arg2-Original	73.04	57.44	64.31
Arg2-Mapped	84.11	60.55	70.41

Table 3: SRL System Performance on Arg1 Mapping and Arg2 Mapping, tested using the *WSJ corpus (section 23)*. This represents performance on the same genre as the training corpus.

System	Precision	Recall	F1
Arg1-Original	86.01	71.46	78.07
Arg1-Mapped	88.24	71.15	78.78
Arg2-Original	66.74	52.22	58.59
Arg2-Mapped	81.45	58.45	68.06

Table 4: SRL System Performance on Arg1 Mapping and Arg2 Mapping, tested using the *PropBanked Brown corpus*. This represents performance on a different genre from the training corpus.

Confusion Matrix		ARG2-Original		
		ARG1	ARG2	ARGM
ARG2-Mapped	ARG0	53	50	-
	ARG1	-	716	-
	ARG2	1	-	2
	ARG3	-	1	-
	ARGM	1	482	-
233 ARG2-Mapped arguments are not labeled by ARG2-Original				

Table 5: Confusion matrix on the 1,539 instances which ARG2-Mapped tags correctly and ARG2-Original fails to predict.

thematic role groups, system performance was evaluated based solely on the PropBank role label that was assigned.

We had hypothesized that with the use of thematic roles, we would be able to create a more consistent training data set which would result in an improvement in system performance. In addition, the thematic roles would behave more consistently than the overloaded Args[2-5] across verbs, which should enhance robustness. However, since in practice we are also increasing the number of argument labels an SRL system needs to tag, the system might suffer from data sparseness. Our hope is that the enhancement gained from the mapping will outweigh the loss due to data sparseness.

From Table 3 and Table 4 we see the F1 scores of Arg1-Original and Arg1-Mapped are statistically indifferent both on the WSJ corpus and the Brown corpus. These results confirm the observation that Arg1 in the PropBank behaves fairly verb-independently so that the VerbNet mapping does not provide much benefit. The increase of precision due to a more coherent training data set is compensated for by the loss of recall due to data sparseness.

The results of the Arg2 experiments tell a different story. Both precision and recall are improved significantly, which demonstrates that the Arg2 label in the PropBank is quite overloaded. The Arg2 mapping improves the overall results (F1) on the WSJ by 6% and on the Brown corpus by almost 10%. As a more diverse corpus, the Brown corpus provides many more opportunities for generalizing to new usages. Our new SRL system handles these cases more robustly, demonstrating the consistency and usefulness of the thematic role categories.

5.4 Improved Argument Distinction via Mapping

The ARG2-Mapped system generalizes well both on the WSJ corpus and the Brown corpus. In order to explore the improved robustness brought by the mapping, we extracted and observed the 1,539 instances to which the system ARG2-Mapped assigned the correct semantic role label, but which the system ARG2-Original failed to predict. From the confusion matrix depicted in Table 5, we discover the following:

The mapping makes ARG2 more clearly defined, and as a result there is a better distinction between ARG2 and other argument labels: Among the 1,539 instances that ARG2-Original didn't tag correctly, 233 instances are not assigned an argument label, and 1,252 instances ARG2-Original confuse the ARG2 label with another argument label: the system ARG2-Original assigned the ARG2 label to 50 ARG0's, 716 ARG1's, 1 ARG3 and 482 ARGM's, and assigned other argument labels to 3 ARG2's.

6 Conclusions

In conclusion, we have described a mapping from the annotated PropBank corpus to VerbNet verb classes with associated thematic role labels. We hypothesized that these labels would be more verb-independent and less overloaded than the PropBank Args2-5, and would therefore provide more consistent training instances which would generalize better to new genres. Our preliminary experiments confirm this hypothesis, with a 6% performance improvement on the WSJ and a 10% performance improvement on the Brown corpus for Arg2.

In future work, we will map the PropBank-ed Brown corpus to VerbNet as well, which will allow much more thorough testing of our hypothesis. We will also examine back-off to verb class membership as a technique for improving performance on out of vocabulary verbs. Finally, we plan to explore the effect of different thematic role groupings on system performance.

References

Xavier Carreras and Lluís Márquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

John Chen and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of EMNLP-2003*, Sapporo, Japan.

D. R. Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67:574–619.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 57–64, Sapporo, Japan.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2003. Shallow semantic parsing using support vector machines. Technical report, The Center for Spoken Language Research at the University of Colorado (CSLR).

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press.

Edward Loper, Szu-ting Yi, and Martha Palmer. 2007. Empirical evidence for useful semantic role categories. In *Proceedings of the International Workshop on Computational Linguistics*.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotating predicate argument structure.

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *Proceedings of the 42-th Conference on Association for Computational Linguistic (ACL-2004)*, Barcelona, Spain.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1):71–106.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, H. Martin, James, and Daniel Jurafsky. 2005a. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL-2005*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005)*, Ann Arbor, MI.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Kristina Toutanova, Aria Haghighi, and Christopher D. 2005. Joint learning improves semantic role labeling. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005)*, Ann Arbor, MI.

Szu-ting Yi and Martha Palmer. 2004. Pushing the boundaries of semantic role labeling with svm. In *Proceedings of the International Conference on Natural Language Processing*.

Szu-ting Yi and Martha Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.

Towards Robust Semantic Role Labeling

Sameer Pradhan
BBN Technologies
Cambridge, MA 02138
pradhan@bbn.com

Wayne Ward, James H. Martin
University of Colorado
Boulder, CO 80303
{whw,martin}@colorado.edu

Abstract

Most research on semantic role labeling (SRL) has been focused on training and evaluating on the same corpus in order to develop the technology. This strategy, while appropriate for initiating research, can lead to over-training to the particular corpus. The work presented in this paper focuses on analyzing the robustness of an SRL system when trained on one genre of data and used to label a different genre. Our state-of-the-art semantic role labeling system, while performing well on WSJ test data, shows significant performance degradation when applied to data from the Brown corpus. We present a series of experiments designed to investigate the source of this lack of portability. These experiments are based on comparisons of performance using PropBanked WSJ data and PropBanked Brown corpus data. Our results indicate that while syntactic parses and argument identification port relatively well to a new genre, argument classification does not. Our analysis of the reasons for this is presented and generally point to the nature of the more lexical/semantic features dominating the classification task and general structural features dominating the argument identification task.

1 Introduction

Automatic, accurate and wide-coverage techniques that can annotate naturally occurring text with semantic argument structure play a key role in NLP applications such as Information Extraction (Surdeanu et al., 2003; Harabagiu et al., 2005), Question Answering (Narayanan and Harabagiu, 2004) and Machine Translation (Boas, 2002; Chen and Fung, 2004). Semantic Role Labeling (SRL) is the process of producing such a markup. When presented with a sentence, a parser should, for each predicate in the sentence, identify and label the predicate's semantic arguments. In recent work, a number of researchers have cast this problem as a tagging problem and have applied various supervised machine learning techniques to it. On the Wall Street Journal (WSJ) data, using correct syntactic parses, it is possible to achieve accuracies rivaling human inter-annotator agreement. However, the performance gap widens when information derived from automatic syntactic parses is used.

So far, most of the work on SRL systems has been focused on improving the labeling performance on a test set belonging to the same genre of text as the training set. Both the Treebank on which the syntactic parser is trained and the PropBank on which the SRL systems are trained represent articles from the year 1989 of the WSJ. While all these systems perform quite well on the WSJ test data, they show significant performance degradation (approximately 10 point drop in F-score) when applied to label test data that is different than the genre that WSJ represents (Pradhan et al., 2004; Carreras and Màrquez, 2005).

Surprisingly, it does not matter much whether the data is from another newswire, or a completely different type of text – as in the Brown corpus. These results indicate that the systems are being over-fit to the specific genre of text. Many performance improvements on the WSJ PropBank corpus may reflect tuning to the corpus. For the technology to be widely accepted and useful, it must be robust to change in genre of the data. Until recently, data tagged with similar semantic argument structure was not available for multiple genres of text. Recently, Palmer et al., (2005), have PropBanked a significant portion of the Treebanked Brown corpus which enables us to perform experiments to analyze the reasons behind the performance degradation, and suggest potential solutions.

2 Semantic Annotation and Corpora

In the PropBank[1] corpus (Palmer et al., 2005), predicate argument relations are marked for the verbs in the text. PropBank was constructed by assigning semantic arguments to constituents of the hand-corrected Treebank parses. The arguments of a verb are labeled ARG0 to ARG5, where ARG0 is the PROTO-AGENT (usually the subject of a transitive verb) ARG1 is the PROTO-PATIENT (usually its direct object), etc. In addition to these CORE ARGUMENTS, 16 additional ADJUNCTIVE ARGUMENTS, referred to as ARGMs are also marked.

More recently the PropBanking effort has been extended to encompass multiple corpora. In this study we use PropBanked versions of the Wall Street Journal (WSJ) part of the Penn Treebank (Marcus et al., 1994) and part of the Brown portion of the Penn Treebank.

The WSJ PropBank data comprise 24 sections of the WSJ, each section representing about 100 documents. PropBank release 1.0 contains about 114,000 predicates instantiating about 250,000 arguments and covering about 3,200 verb lemmas. Section 23, which is a standard test set and a test set in some of our experiments, comprises 5,400 predicates instantiating about 12,000 arguments.

The Brown corpus is a Standard Corpus of American English that consists of about one million words of English text printed in the calendar year 1961 (Kučera and Francis, 1967). The corpus contains about 500 samples of 2000+ words each. The idea behind creating this corpus was to create a heterogeneous sample of English text so that it would be useful for comparative language studies.

The Release 3 of the Penn Treebank contains the hand parsed syntactic trees of a subset of the Brown Corpus – sections F, G, K, L, M, N, P and R. Palmer et al., (2005) have recently PropBanked a significant portion of this Treebanked Brown corpus. In all, about 17,500 predicates are tagged with their semantic arguments. For these experiments we used a limited release of PropBank dated September 2005. A small portion of the predicates – about 8,000 have also been tagged with frame sense information.

3 SRL System Description

We formulate the labeling task as a classification problem as initiated by Gildea and Jurafsky (2002) and use Support Vector Machine (SVM) classifiers (2005). We use TinySVM[2] along with YamCha[3] (Kudo and Matsumoto, 2000) (Kudo and Matsumoto, 2001) as the SVM training and classification software. The system uses a polynomial kernel with degree 2; the cost per unit violation of the margin, C=1; and, tolerance of the termination criterion, e=0.001. More details of this system can be found in Pradhan et al., (2005). The performance of this system on section 23 of the WSJ when trained on sections 02-21 is shown in Table 1

ALL ARGS	Task	P	R	F	A
		(%)	(%)		(%)
TREEBANK	Id.	97.5	96.1	96.8	
	Class.	-	-	-	93.0
	Id. + Class.	91.8	90.5	91.2	
AUTOMATIC	Id.	86.9	84.2	85.5	
	Class.	-	-	-	92.0
	Id. + Class.	82.1	77.9	79.9	

Table 1: Performance of the SRL system on WSJ

The performance of the SRL system is reported on three different tasks, all of which are with respect to a particular predicate: i) *argument identification* (ID), is the task of identifying the set of words (here, parse constituents) that represent a semantic role; ii) *argument classification* (Class.), is the task of classifying parse constituents known to represent some

[1] http://www.cis.upenn.edu/~ace/

[2] http://cl.aist-nara.ac.jp/~talus-Au/software/TinySVM/

[3] http://cl.aist-nara.ac.jp/~taku-Au/software/yamcha/

semantic role into one of the many semantic role types; and iii) *argument identification and classification* (ID + Class.), which involves both the identification of the parse constituents that represent semantic roles of the predicate and their classification into the respective semantic roles. As usual, argument classification is measured as percent accuracy (A), whereas ID and ID + Class. are measured in terms of precision (P), recall (R) and F-score (F) – the harmonic mean of P and R. The first three rows of Table 1 report performance for the system that uses hand-corrected Treebank parses, and the next three report performance for the SRL system that uses automatically generated – Charniak parser – parses, both during training and testing.

4 Robustness Experiments

This section describes experiments that we performed using the PropBanked Brown corpus in an attempt to analyze the factors affecting the portability of SRL systems.

4.1 How does the SRL system trained on WSJ perform on Brown?

In order to test the robustness of the SRL system, we used a system trained on the PropBanked WSJ corpus to label data from the Brown corpus. We use the entire PropBanked Brown corpus (about 17,500 predicates) as a test set for this experiment and use the SRL system trained on WSJ sections 02-21 to tag its arguments.

Table 2 shows the performance for training and testing on WSJ, and for training on WSJ and testing on Brown. There is a significant reduction in performance when the system trained on WSJ is used to label data from the Brown corpus. The degradation in the Identification task is small compared to that of the combined Identification and Classification task. A number of factors could be responsible for the loss of performance. It is possible that the SRL models are tuned to the particular vocabulary and sense structure associated with the training data. Also, since the syntactic parser that is used for generating the syntax parse trees (Charniak) is heavily lexicalized and is trained on WSJ, it could have decreased accuracy on the Brown data resulting in reduced accuracy for Semantic Role Labeling. Since the SRL algorithm walks the syntax tree classifying each node, if no constituent node is present that corresponds to the correct argument, the system cannot produce a correct labeling for the argument.

Train	Test	Id.	Id. + Class
		F	F
WSJ	WSJ	85.5	79.9
WSJ	Brown	82.4	65.1

Table 2: Performance of the SRL system on Brown.

In order to check the extent to which constituent nodes representing semantic arguments were deleted from the syntax tree due to parser error, we generated the performance numbers which are shown in Table 3. These numbers are for top one parse for the Charniak parser, and represent not all parser errors, but deletion of argument bearing constituent nodes.

	Total	Misses	%
PropBank	12000	800	6.7
Brown	45880	3692	8.1

Table 3: Constituent deletions in WSJ and Brown.

The parser misses 6.7% of the argument-bearing nodes in the PropBank test set and about 8.1% in the Brown corpus. This indicates that the errors in syntactic parsing account for a fairly small amount of the argument deletions and probably do not contributing significantly to the increased SRL error rate. Obviously, just the presence of a argument-bearing constituent does not necessarily guarantee the correctness of the structural connections between itself and the predicate.

4.2 Identification vs Classification Performance

Different features tend to dominate in the identification task vs the classification task. For example, the path feature (representing the path in the syntax tree from the argument to the predicate) is the single most salient feature for the ID task and is not very important in the classification task. In the next experiment we look at cross genre performance of the ID and Classification tasks. We used gold standard syntactic trees from the Treebank so there are no errors in generating the syntactic structure. In addition to training on the WSJ and testing on WSJ and Brown, we trained the SRL system on a Brown training set and tested it on a test set also from the Brown corpus. In generating the Brown training and

SRL Train	SRL Test	Task	P (%)	R (%)	F	A (%)
WSJ (104k)	WSJ (5k)	Id.	97.5	96.1	96.8	
		Class.				93.0
		Id. + Class.	91.8	90.5	91.2	
WSJ (14k)	WSJ (5k)	Id.	96.3	94.4	95.3	
		Class.				86.1
		Id. + Class.	84.4	79.8	82.0	
BROWN (14k)	BROWN (1.6k)	Id.	95.7	94.9	95.2	
		Class.				80.1
		Id. + Class.	79.9	77.0	78.4	
WSJ (14k)	BROWN (1.6k)	Id.	94.2	91.4	92.7	
		Class.				72.0
		Id. + Class.	71.8	65.8	68.6	

Table 4: Performance of the SRL system using correct Treebank parses.

test sets, we used stratified sampling, which is often used by the syntactic parsing community (Gildea, 2001). The test set was generated by selecting every 10^{th} sentence in the Brown Corpus. We also held out the development set used by Bacchiani et al., (2006) to tune system parameters in the future. This procedure resulted in a training set of approximately 14,000 predicates and a test set of about 1600 predicates. We did not perform any parameter tuning for any of the following experiments, and used the parameter settings from the best performing version of the SRL system as reported in Table1. We compare the performance on this test set with that obtained when the SRL system is trained using WSJ sections 02-21 and use section 23 for testing. For a more balanced comparison, we retrained the SRL system on the same amount of data as used for training on Brown, and tested it on section 23. As usual, trace information, and function tag information from the Treebank is stripped out.

Table 4 shows the results. There is a fairly small difference in argument Identification performance when the SRL system is trained on 14,000 predicates vs 104,000 predicates from the WSJ (F-score 95.3 vs 96.8). However, there is a considerable drop in Classification accuracy (86.1% vs 93.0%). When the SRL system is trained and tested on Brown data, the argument Identification performance is not significantly different than that for the system trained and tested on WSJ data (F-score 95.2 vs 95.3). The drop in argument Classification accuracy is much more severe (86.1% vs 80.1%).

This same trend between ID and Classification is even more pronounced when training on WSJ and testing on Brown. For a system trained on WSJ, there is a fairly small drop in performance of the ID task when tested on Brown vs tested on WSJ (F-score 92.7 vs 95.3). However, in this same condition, the Classification task has a very large drop in performance (72.0% vs 86.1%).

So argument ID is not very sensitive to amount of training data in a corpus, or to the genre of the corpus, and ports well from WSJ to Brown. This experiment supports the belief that there is no significant drop in the task of identifying the right syntactic constituents that are arguments – and this is intuitive since previous experiments have shown that the task of argument identification is more dependent on the structural features – one such feature being the path in the syntax tree.

Argument Classification seems to be the problem. It requires more training data within the WSJ corpus, does not perform as well when trained and tested on Brown as it does for WSJ and does not port well from WSJ to Brown. This suggests that the features it uses are being over-fit to the training data and are more idiosyncratic to a given dataset. In particular, the predicate whose arguments are being identified, and the head word of the syntactic constituent being classified are both important features in the task of argument classification.

As a generalization, the features used by the Identification task reflect structure and port well. The features used by the Classification task reflect specific lexical usage and semantics, and tend to require more training data and are more subject to over-fitting. Even when training and testing on Brown, Classification accuracy is considerably worse than training and testing on WSJ (with comparable training set size). It is probably the case that the predicates and head words in a homogeneous corpus such as the WSJ are used more consistently, and tend to have single dominant word senses. The Brown corpus probably has much more variety in its lexical usage and word senses.

4.3 How sensitive is semantic argument prediction to the syntactic correctness across genre?

This experiment examines the same cross-genre effects as the last experiment, but uses automatically generated syntactic parses rather than gold standard ones.

For this experiment, we used the same amount of training data from WSJ as available in the Brown training set – that is about 14,000 predicates. The examples from WSJ were selected randomly. The Brown test set is the same as used in the previous experiment, and the WSJ test set is the entire section 23.

Recently there have been some improvements to the Charniak parser, use *n*-best re-ranking as reported in (Charniak and Johnson, 2005) and self-training and re-ranking using data from the North American News corpus (NANC) and adapts much better to the Brown corpus (McClosky et al., 2006a; McClosky et al., 2006b). The performance of these parsers as reported in the respective literature are shown in Table 6 shows the performance (as reported in the literature) of the Charniak parser: when trained and tested on WSJ, when trained on WSJ and tested on Brown, When trained and tested on Brown, and when trained on WSJ and adapted with NANC.

Train	Test	F
WSJ	WSJ	91.0
WSJ	Brown	85.2
Brown	Brown	88.4
WSJ+NANC	Brown	87.9

Table 6: Charniak parser performance.

We describe the results of Semantic Role Labeling under the following five conditions:

1. The SRL system is trained on features extracted from automatically generated parses of the PropBanked WSJ sentences. The syntactic parser – Charniak parser – is itself trained on the WSJ training sections of the Treebank. This is used for Semantic Role Labeling of section-23 of WSJ.
2. The SRL system is trained on features extracted from automatically generated parses of the PropBanked WSJ sentences. The syntactic parser – Charniak parser – is itself trained on the WSJ training sections of the Treebank. This is used to classify the Brown test set.
3. The SRL system is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is trained using the WSJ portion of the Treebank. This is used to classify the Brown test set.
4. The SRL system is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is trained using the Brown training portion of the Treebank. This is used to classify the Brown test set.
5. The SRL system is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is the version that is self-trained using 2,500,000 sentences from NANC, and where the starting version is trained only on WSJ data (McClosky et al., 2006b). This is used to classify the Brown test set.

Table 5 shows the results. For simplicity of discussion we have tagged the five conditions as 1., 2., 3., 4., and 5. Comparing conditions 2. and 3. shows that when the features used to train the SRL system are extracted using a syntactic parser that is trained on WSJ it performs at almost the same level on the task of Identification, regardless of whether it is trained on the PropBanked Brown corpus or the PropBanked WSJ corpus. This, however, is significantly lower than when all the three – the syntactic parser training set, the SRL system training set, and the SRL system test set, are from the same genre (6 F-score points lower than condition 1, and 5 points lower than conditions 4 and 5). In case of the combined task, the gap between the performance for conditions 2 and 3 is about 10 points in F-score (59.1 vs 69.8). Looking at the argument classification accuracies, we see that using the SRL system

Setup	Parser Train	SRL Train	SRL Test	Task	P (%)	R (%)	F	A (%)
1.	WSJ	WSJ	WSJ	Id.	87.3	84.8	86.0	
	(40k – sec:00-21)	(14k)	(5k)	Class.				84.1
				Id. + Class.	77.5	69.7	73.4	
2.	WSJ	WSJ	Brown	Id.	81.7	78.3	79.9	
	(40k – sec:00-21)	(14k)	(1.6k)	Class.				72.1
				Id. + Class.	63.7	55.1	59.1	
3.	WSJ	Brown	Brown	Id.	81.7	78.3	80.0	
	(40k – sec:00-21)	(14k)	(1.6k)	Class.				79.2
				Id. + Class.	78.2	63.2	69.8	
4.	Brown	Brown	Brown	Id.	87.6	82.3	84.8	
	(20k)	(14k)	(1.6k)	Class.				78.9
				Id. + Class.	77.4	62.1	68.9	
5.	WSJ+NANC	Brown	Brown	Id.	87.7	82.5	85.0	
	(2,500k)	(14k)	(1.6k)	Class.				79.9
				Id. + Class.	77.2	64.4	70.0	

Table 5: Performance on WSJ and Brown using automatic syntactic parses

trained on WSJ to test Brown sentences give a 12 point drop in F-score (84.1 vs 72.1). Using the SRL system trained on Brown using WSJ trained syntactic parser shows a drop in accuracy by about 5 F-score points (84.1 to 79.2). When the SRL system is trained on Brown using syntactic parser also trained on Brown, we get a quite similar classification performance, which is again about 5 points lower than what we get using all WSJ data. This shows lexical semantic features might be very important to get a better argument classification on Brown corpus.

4.4 How much data is required to adapt to a new genre?

We would like to know how much data from a new genre we need to annotate and add to the training data of an existing corpus to adapt the system such that it gives the same level of performance as when it is trained on the new genre.

One section of the Brown corpus – section CK has about 8,200 predicates annotated. We use six different conditions – two in which we use correct Treebank parses, and the four others in which we use automatically generated parses using the variations described before. All training sets start with the same number of examples as in the Brown training set. The part of this section used as a test set for the CoNLL 2005 shared task is used as the test set here. It contains a total of about 800 predicates.

Table 7 shows a comparison of these conditions. In all the six conditions, the performance on the task of Identification and Classification improves gradually until about 5625 examples of section CK which is about 75% of the total added, above which they improve very little. In fact, even 50% of the new data accounts for 90% of the performance difference. Even when the syntactic parser is trained on WSJ and the SRL is trained on WSJ, adding 7,500 instances of the new genres allows it to achieve almost the same performance as when all three are from the same genre (67.2 vs 69.9). Numbers for argument identification aren't shown because adding more data does not have any statistically significant impact on its performance. The system that uses self-trained syntactic parser seems to perform slightly better than the rest of the versions that use automatically generated syntactic parses. The precision numbers are almost unaffected – except when the labeler is trained on WSJ PropBank data.

4.5 How much does verb sense information contribute?

In order to find out how important the verb sense information is in the process of genre transfer, we used the subset of PropBanked Brown corpus that was tagged with verb sense information, ran an experiment similar to that of Experiment 1. We used the oracle sense information and correct syntactic information for this experiment.

Table 8 shows the results of this experiment. There is about 1 point F-score increase on using oracle sense information on the overall data. We looked at predicates that had high perplexity in both the training and test sets, and whose sense distribu-

Parser	SRL	Id. + Class			Parser	SRL	Id. + Class		
		P	R	F			P	R	F
Train	Train	(%)	(%)				(%)	(%)	
WSJ	WSJ (14k)				WSJ	Brown (14k)			
(Treebank parses)	(Treebank parses)								
	+0 ex. from CK	74.1	66.5	70.1	(40k)	+0 ex. from CK	74.4	57.0	64.5
	+1875 ex. from CK	77.6	71.3	74.3		+1875 ex. from CK	75.1	58.7	65.9
	+3750 ex. from CK	79.1	74.1	76.5		+3750 ex. from CK	76.1	59.6	66.9
	+5625 ex. from CK	80.4	76.1	78.1		+5625 ex. from CK	76.9	60.5	67.7
	+7500 ex. from CK	80.2	76.1	78.1		+7500 ex. from CK	76.8	59.8	67.2
Brown	Brown (14k)				Brown	Brown (14k)			
(Treebank parses)	(Treebank parses)								
	+0 ex. from CK	77.1	73.0	75.0	(20k)	+0 ex. from CK	76.0	59.2	66.5
	+1875 ex. from CK	78.8	75.1	76.9		+1875 ex. from CK	76.1	60.0	67.1
	+3750 ex. from CK	80.4	76.9	78.6		+3750 ex. from CK	77.7	62.4	69.2
	+5625 ex. from CK	80.4	77.2	78.7		+5625 ex. from CK	78.2	63.5	70.1
	+7500 ex. from CK	81.2	78.1	79.6		+7500 ex. from CK	78.2	63.2	69.9
WSJ	WSJ (14k)				WSJ+NANC	Brown (14k)			
(40k)	+0 ex. from CK	65.2	55.7	60.1	(2,500k)	+0 ex. from CK	74.4	60.1	66.5
	+1875 ex. from CK	68.9	57.5	62.7		+1875 ex. from CK	76.2	62.3	68.5
	+3750 ex. from CK	71.8	59.3	64.9		+3750 ex. from CK	76.8	63.6	69.6
	+5625 ex. from CK	74.3	61.3	67.2		+5625 ex. from CK	77.7	63.8	70.0
	+7500 ex. from CK	74.8	61.0	67.2		+7500 ex. from CK	78.2	64.9	70.9

Table 7: Effect of incrementally adding data from a new genre

Train	Test	Without Sense Id. F	With Sense Id. F
WSJ	Brown (All)	69.1	69.9
WSJ	Brown (predicate: go)	46.9	48.9

Table 8: Influence of verb sense feature.

tion was different. One such predicate is "go". The improvement on classifying the arguments of this predicate was about 2 points (46.9 to 48.9), which suggests that verb sense is more important when the sense structure of the test corpus is more ambiguous and is different from the training. Here we used oracle verb sense information, but one can train a classifier as done by Girju et al., (2005) which achieves a disambiguation accuracy in the 80s for within the WSJ corpus.

5 Conclusions

Our experimental results on robustness to change in genre can be summarized as follows:

- There is a significant drop in performance when training and testing on different corpora – for both Treebank and Charniak parses
- In this process the classification task is more disrupted than the identification task.
- There is a performance drop in classification even when training and testing on Brown (compared to training and testing on WSJ)
- The syntactic parser error is not a large part of the degradation for the case of automatically generated parses.

An error analysis leads us to believe that some reasons for this behavior could be: i) lexical usages that are specific to WSJ, ii) variation in subcategorization across corpora, iii) variation in word sense distribution and iv) changes in topics and entities. Training and testing on the same corpora tends to give a high weight to very specific semantic features. Two possibilities remedies could be: i) using less homogeneous corpora and ii) less specific features, for eg., proper names are replaced with the name entities that they represent. This way the system could be forced to use the more general features. Both of these manipulations would most likely reduce performance on the training set, and on test sets of the same genre as the training data. But they would be likely to generalize better.

6 Acknowledgments

We are extremely grateful to Martha Palmer for providing us with the PropBanked Brown corpus, and to David McClosky for providing us with hypotheses on the Brown test set as well as a cross-validated

version of the Brown training data for the various models reported in his work reported at HLT 2006.

This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grants IS-9978025 and ITR/HCI 0086132.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Hans Boas. 2002. Bilingual framenet dictionaries for machine translation. In *Proceedings of LREC-2002*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, pages 152–164, Ann Arbor, MI.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fi ne n-best parsing and maxent discriminative reranking. In *Proceedings of ACL-2005*, pages 173–180, Ann Arbor, MI.

Benfeng Chen and Pascale Fung. 2004. Automatic construction of an english-chinese bilingual framenet. In *Proceedings of the HLT/NAACL-2004*, Boston, MA.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea. 2001. Corpus variation and parser performance. In *In Proceedings of EMNLP-2001*.

R. Girju, D. Roth, and M. Sammons. 2005. Token-level disambiguation of verbnet classes. In *Proceedings of the Interdisciplinary Workshop on the Identifi -cation and Representation of Verb Features and Verb Classes, K. Erk, A. Melinger, and S. Schulte im Walde (eds.)*.

Sanda Harabagiu, Cosmin Adrian Bejan, and Paul Morarescu. 2005. Shallow semantics for relation extraction. In *IJCAI-2005*, pages 1061–1067, Edinburgh, Scotland.

Henry Kučera and W. Nelson Francis. 1967. *Computational analysis of present-day American English*. Brown University Press, Providence, RI.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identifi cation. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 142–144.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the NAACL-2001*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of HLT/NAACL-2006*, pages 152–159, New York City, USA. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Rerankinng and self-training for parser adaptation. In *Proceedings of COLING/ACL-2006*, Sydney, Australia.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of COLING-2004)*, Geneva, Switzerland.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL-2004*, Boston, MA.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*, Ann Arbor, MI.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*, Sapporo, Japan.

ISP: Learning Inferential Selectional Preferences

Patrick Pantel[†], Rahul Bhagat[†], Bonaventura Coppola[‡], Timothy Chklovski[†], Eduard Hovy[†]

[†]Information Sciences Institute
University of Southern California
Marina del Rey, CA
{pantel,rahul,timc,hovy}@isi.edu

[‡]ITC-Irst and University of Trento
Via Sommarive, 18 – Povo 38050
Trento, Italy
coppolab@itc.it

Abstract

Semantic inference is a key component for advanced natural language understanding. However, existing collections of automatically acquired inference rules have shown disappointing results when used in applications such as textual entailment and question answering. This paper presents *ISP*, a collection of methods for automatically learning admissible argument values to which an inference rule can be applied, which we call *inferential selectional preferences*, and methods for filtering out incorrect inferences. We evaluate *ISP* and present empirical evidence of its effectiveness.

1 Introduction

Semantic inference is a key component for advanced natural language understanding. Several important applications are already relying heavily on inference, including question answering (Moldovan et al. 2003; Harabagiu and Hickl 2006), information extraction (Romano et al. 2006), and textual entailment (Szpektor et al. 2004).

In response, several researchers have created resources for enabling semantic inference. Among manual resources used for this task are WordNet (Fellbaum 1998) and Cyc (Lenat 1995). Although important and useful, these resources primarily contain *prescriptive* inference rules such as "*X divorces Y* $\Rightarrow$ *X married Y*". In practical NLP applications, however, *plausible* inference rules such as "*X* married *Y*" $\Rightarrow$ "*X* dated *Y*" are very useful. This, along with the difficulty and labor-intensiveness of generating exhaustive lists of rules, has led researchers to focus on automatic methods for building inference resources such as inference rule collections (Lin and Pantel 2001; Szpektor et al. 2004) and paraphrase collections (Barzilay and McKeown 2001).

Using these resources in applications has been hindered by the large amount of incorrect inferences they generate, either because of altogether incorrect rules or because of blind application of plausible rules without considering the context of the relations or the senses of the words. For example, consider the following sentence:

Terry Nichols was charged by federal prosecutors for murder and conspiracy in the Oklahoma City bombing.

and an inference rule such as:

X is charged by Y $\Rightarrow$ *Y announced the arrest of X* (1)

Using this rule, we can infer that "*federal prosecutors* announced the arrest of *Terry Nichols*". However, given the sentence:

Fraud was suspected when accounts were charged by CCM telemarketers without obtaining consumer authorization.

the plausible inference rule (1) would incorrectly infer that "*CCM telemarketers* announced the arrest of *accounts*".

This example depicts a major obstacle to the effective use of automatically learned inference rules. What is missing is knowledge about the admissible argument values for which an inference rule holds, which we call *Inferential Selectional Preferences*. For example, inference rule (1) should only be applied if *X* is a *Person* and *Y* is a *Law Enforcement Agent* or a *Law Enforcement Agency*. This knowledge does not guarantee that the inference rule will hold, but, as we show in this paper, goes a long way toward filtering out erroneous applications of rules.

In this paper, we propose *ISP*, a collection of methods for learning inferential selectional preferences and filtering out incorrect inferences. The presented algorithms apply to any collection of inference rules between binary semantic relations, such as example (1). *ISP* derives inferential selectional preferences by aggregating statistics of inference rule instantiations over a large corpus of text. Within *ISP*, we explore different probabilistic models of selectional preference to accept or reject specific inferences. We present empirical evidence to support the following main contribution:

Claim*: Inferential selectional preferences can be automatically learned and used for effectively filtering out incorrect inferences.*

2 Previous Work

Selectional preference (SP) as a foundation for computational semantics is one of the earliest topics in AI and NLP, and has its roots in (Katz and Fodor 1963). Overviews of NLP research on this theme are (Wilks and Fass 1992), which includes the influential theory of Preference Semantics by Wilks, and more recently (Light and Greiff 2002).

Rather than venture into learning inferential SPs, much previous work has focused on learning SPs for simpler structures. Resnik (1996), the seminal paper on this topic, introduced a statistical model for learning SPs for predicates using an unsupervised method.

Learning SPs often relies on an underlying set of *semantic classes*, as in both Resnik's and our approach. Semantic classes can be specified manually or derived automatically. Manual collections of semantic classes include the hierarchies of WordNet (Fellbaum 1998), Levin verb classes (Levin 1993), and FrameNet (Baker et al. 1998). Automatic derivation of semantic classes can take a variety of approaches, but often uses corpus methods and the Distributional Hypothesis (Harris 1964) to automatically cluster similar entities into classes, e.g. CBC (Pantel and Lin 2002). In this paper, we experiment with two sets of semantic classes, one from WordNet and one from CBC.

Another thread related to our work includes extracting from text corpora paraphrases (Barzilay and McKeown 2001) and inference rules, e.g. TEASE[1] (Szpektor et al. 2004) and DIRT (Lin and Pantel 2001). While these systems differ in their approaches, neither provides for the extracted inference rules to hold or fail based on SPs. Zanzotto et al. (2006) recently explored a different interplay between SPs and inferences. Rather than examine the role of SPs in inferences, they use SPs of a particular type to derive inferences. For instance the preference of *win* for the subject *player*, a nominalization of *play*, is used to derive that "win $\Rightarrow$ play". Our work can be viewed as complementary to the work on extracting semantic inferences and paraphrases, since we seek to refine when a given inference applies, filtering out incorrect inferences.

[1] Some systems refer to inferences they extract as *entailments*; the two terms are sometimes used interchangeably.

3 Selectional Preference Models

The aim of this paper is to learn inferential selectional preferences for filtering inference rules.

Let $p_i \Rightarrow p_j$ be an inference rule where p is a binary semantic relation between two entities x and y. Let $\langle x, p, y\rangle$ be an instance of relation p.

Formal task definition*: Given an inference rule* $p_i \Rightarrow p_j$ *and the instance* $\langle x, p_i, y\rangle$*, our task is to determine if* $\langle x, p_j, y\rangle$ *is valid.*

Consider the example in Section 1 where we have the inference rule "*X is charged by Y*" $\Rightarrow$ "*Y announced the arrest of X*". Our task is to automatically determine that "*federal prosecutors* announced the arrest of *Terry Nichols*" (i.e., $\langle$*Terry Nichols*, p_j, *federal prosecutors*$\rangle$) is valid but that "*CCM telemarketers* announced the arrest of *accounts*" is invalid.

Because the semantic relations p are binary, the selectional preferences on their two arguments may be either considered jointly or independently. For example, the relation p = "*X is charged by Y*" could have joint SPs:

$\langle$*Person, Law Enforcement Agent*$\rangle$
$\langle$*Person, Law Enforcement Agency*$\rangle$ (2)
$\langle$*Bank Account, Organization*$\rangle$

or independent SPs:

$\langle$*Person, **$\rangle$
$\langle$**, Organization*$\rangle$ (3)
$\langle$**, Law Enforcement Agent*$\rangle$

This distinction between joint and independent selectional preferences constitutes the difference between the two models we present in this section.

The remainder of this section describes the *ISP* approach. In Section 3.1, we describe methods for automatically determining the semantic contexts of each single relation's selectional preferences. Section 3.2 uses these for developing our inferential selectional preference models. Finally, we propose inference filtering algorithms in Section 3.3.

3.1 Relational Selectional Preferences

Resnik (1996) defined the selectional preferences of a predicate as the semantic classes of the words that appear as its arguments. Similarly, we define the *relational selectional preferences* of a binary semantic relation p_i as the semantic classes $C(x)$ of the words that can be instantiated for x and as the semantic classes $C(y)$ of the words that can be instantiated for y.

The semantic classes $C(x)$ and $C(y)$ can be obtained from a conceptual taxonomy as proposed in (Resnik 1996), such as WordNet, or from the classes extracted from a word clustering algorithm such as CBC (Pantel and Lin 2002). For example, given the relation "*X is charged by Y*", its relational selection preferences from WordNet could be {*social_group, organism, state…*} for *X* and {*authority, state, section…*} for *Y*.

Below we propose joint and independent models, based on a corpus analysis, for automatically determining relational selectional preferences.

Model 1: Joint Relational Model (JRM)

Our joint model uses a corpus analysis to learn SPs for binary semantic relations by considering their arguments jointly, as in example (2).

Given a large corpus of English text, we first find the occurrences of each semantic relation p. For each instance $\langle x, p, y\rangle$, we retrieve the sets $C(x)$ and $C(y)$ of the semantic classes that x and y belong to and accumulate the frequencies of the triples $\langle c(x), p, c(y)\rangle$, where $c(x) \in C(x)$ and $c(y) \in C(y)$[2].

Each triple $\langle c(x), p, c(y)\rangle$ is a candidate selectional preference for p. Candidates can be incorrect when: *a*) they were generated from the incorrect sense of a polysemous word; or *b*) p does not hold for the other words in the semantic class.

Intuitively, we have more confidence in a particular candidate if its semantic classes are closely associated given the relation p. Pointwise mutual information (Cover and Thomas 1991) is a commonly used metric for measuring this association strength between two events e_1 and e_2:

$$pmi(e_1;e_2) = \log \frac{P(e_1,e_2)}{P(e_1)P(e_2)} \tag{3.1}$$

We define our ranking function as the strength of association between two semantic classes, c_x and c_y[3], given the relation p:

$$pmi(c_x|p;\ c_y|p) = \log \frac{P(c_x,c_y|p)}{P(c_x|p)P(c_y|p)} \tag{3.2}$$

Let $|c_x, p, c_y|$ denote the frequency of observing the instance $\langle c(x), p, c(y)\rangle$. We estimate the probabilities of Equation 3.2 using maximum likelihood estimates over our corpus:

$$P(c_x|p) = \frac{|c_x,p,*|}{|*,p,*|} \qquad P(c_y|p) = \frac{|*,p,c_y|}{|*,p,*|} \qquad P(c_x,c_y|p) = \frac{|c_x,p,c_y|}{|*,p,*|} \tag{3.3}$$

Similarly to (Resnik 1996), we estimate the above frequencies using:

$$|c_x,p,*| = \sum_{w \in c_x} \frac{|w,p,*|}{|C(w)|} \quad |*,p,c_y| = \sum_{w \in c_y} \frac{|*,p,w|}{|C(w)|} \quad |c_x,p,c_y| = \sum_{w_1 \in c_x, w_2 \in c_y} \frac{|w_1,p,w_2|}{|C(w_1)| \times |C(w_2)|}$$

where $|x, p, y|$ denotes the frequency of observing the instance $\langle x, p, y\rangle$ and $|C(w)|$ denotes the number of classes to which word w belongs. $|C(w)|$ distributes w's mass equally to all of its senses c_w.

Model 2: Independent Relational Model (IRM)

Because of sparse data, our joint model can miss some correct selectional preference pairs. For example, given the relation

Y announced the arrest of X

we may find occurrences from our corpus of the particular class "*Money Handler*" for *X* and "*Lawyer*" for *Y*, however we may never see both of these classes co-occurring even though they would form a valid relational selectional preference.

To alleviate this problem, we propose a second model that is less strict by considering the arguments of the binary semantic relations independently, as in example (3).

Similarly to JRM, we extract each instance $\langle x, p, y\rangle$ of each semantic relation p and retrieve the set of semantic classes $C(x)$ and $C(y)$ that x and y belong to, accumulating the frequencies of the triples $\langle c(x), p, *\rangle$ and $\langle *, p, c(y)\rangle$, where $c(x) \in C(x)$ and $c(y) \in C(y)$.

All tuples $\langle c(x), p, *\rangle$ and $\langle *, p, c(y)\rangle$ are candidate selectional preferences for p. We rank candidates by the probability of the semantic class given the relation p, according to Equations 3.3.

[2] In this paper, the semantic classes $C(x)$ and $C(y)$ are extracted from WordNet and CBC (described in Section 4.2).

[3] c_x and c_y are shorthand for $c(x)$ and $c(y)$ in our equations.

3.2 Inferential Selectional Preferences

Whereas in Section 3.1 we learned selectional preferences for the arguments of a relation *p*, in this section we learn selectional preferences for the arguments of an inference rule $p_i \Rightarrow p_j$.

Model 1: **Joint Inferential Model (JIM)**

Given an inference rule $p_i \Rightarrow p_j$, our joint model defines the set of inferential SPs as the intersection of the relational SPs for p_i and p_j, as defined in the Joint Relational Model (JRM). For example, suppose relation p_i = "*X is charged by Y*" gives the following SP scores under the JRM:

$\langle Person, p_i, Law\ Enforcement\ Agent \rangle = 1.45$
$\langle Person, p_i, Law\ Enforcement\ Agency \rangle = 1.21$
$\langle Bank\ Account, p_i, Organization \rangle = 0.97$

and that p_j = "*Y announced the arrest of X*" gives the following SP scores under the JRM:

$\langle Law\ Enforcement\ Agent, p_j, Person \rangle = 2.01$
$\langle Reporter, p_j, Person \rangle = 1.98$
$\langle Law\ Enforcement\ Agency, p_j, Person \rangle = 1.61$

The intersection of the two sets of SPs forms the candidate inferential SPs for the inference $p_i \Rightarrow p_j$:

$\langle Law\ Enforcement\ Agent, Person \rangle$
$\langle Law\ Enforcement\ Agency, Person \rangle$

We rank the candidate inferential SPs according to three ways to combine their relational SP scores, using the *minimum*, *maximum*, and *average* of the SPs. For example, for ⟨*Law Enforcement Agent, Person*⟩, the respective scores would be 1.45, 2.01, and 1.73. These different ranking strategies produced nearly identical results in our experiments, as discussed in Section 5.

Model 2: **Independent Inferential Model (IIM)**

Our independent model is the same as the joint model above except that it computes candidate inferential SPs using the Independent Relational Model (IRM) instead of the JRM. Consider the same example relations p_i and p_j from the joint model and suppose that the IRM gives the following relational SP scores for p_i:

$\langle Law\ Enforcement\ Agent, p_i, * \rangle = 3.43$
$\langle *, p_i, Person \rangle = 2.17$
$\langle *, p_i, Organization \rangle = 1.24$

and the following relational SP scores for p_j:

$\langle *, p_j, Person \rangle = 2.87$
$\langle Law\ Enforcement\ Agent, p_j, * \rangle = 1.92$
$\langle Reporter, p_j, * \rangle = 0.89$

The intersection of the two sets of SPs forms the candidate inferential SPs for the inference $p_i \Rightarrow p_j$:

$\langle Law\ Enforcement\ Agent, * \rangle$
$\langle *, Person \rangle$

We use the same *minimum*, *maximum*, and *average* ranking strategies as in JIM.

3.3 Filtering Inferences

Given an inference rule $p_i \Rightarrow p_j$ and the instance $\langle x, p_i, y \rangle$, the system's task is to determine whether $\langle x, p_j, y \rangle$ is valid. Let $C(w)$ be the set of semantic classes $c(w)$ to which word w belongs. Below we present three filtering algorithms which range from the least to the most permissive:

- **ISP.JIM**, accepts the inference $\langle x, p_j, y \rangle$ if the inferential SP $\langle c(x), p_j, c(y) \rangle$ was admitted by the Joint Inferential Model for some $c(x) \in C(x)$ and $c(y) \in C(y)$.
- **ISP.IIM.**$\wedge$, accepts the inference $\langle x, p_j, y \rangle$ if the inferential SPs $\langle c(x), p_j, * \rangle$ AND $\langle *, p_j, c(y) \rangle$ were admitted by the Independent Inferential Model for some $c(x) \in C(x)$ and $c(y) \in C(y)$.
- **ISP.IIM.**$\vee$, accepts the inference $\langle x, p_j, y \rangle$ if the inferential SP $\langle c(x), p_j, * \rangle$ OR $\langle *, p_j, c(y) \rangle$ was admitted by the Independent Inferential Model for some $c(x) \in C(x)$ and $c(y) \in C(y)$.

Since both JIM and IIM use a ranking score in their inferential SPs, each filtering algorithm can be tuned to be more or less strict by setting an acceptance threshold on the ranking scores or by selecting only the top τ percent highest ranking SPs. In our experiments, reported in Section 5, we tested each model using various values of τ.

4 Experimental Methodology

This section describes the methodology for testing our claim that inferential selectional preferences can be learned to filter incorrect inferences.

Given a collection of inference rules of the form $p_i \Rightarrow p_j$, our task is to determine whether a particular instance $\langle x, p_j, y \rangle$ holds given that $\langle x, p_i, y \rangle$ holds[4]. In the next sections, we describe our collection of inference rules, the semantic classes used for forming selectional preferences, and evaluation criteria for measuring the filtering quality.

[4] Recall that the inference rules we consider in this paper are not necessary strict logical inference rules, but plausible inference rules; see Section 3.

4.1 Inference Rules

Our models for learning inferential selectional preferences can be applied to any collection of inference rules between binary semantic relations. In this paper, we focus on the inference rules contained in the DIRT resource (Lin and Pantel 2001). DIRT consists of over 12 million rules which were extracted from a 1GB newspaper corpus (San Jose Mercury, Wall Street Journal and AP Newswire from the TREC-9 collection). For example, here are DIRT's top 3 inference rules for "*X solves Y*":

"*Y is solved by X*", "*X resolves Y*", "*X finds a solution to Y*"

4.2 Semantic Classes

The choice of semantic classes is of great importance for selectional preference. One important aspect is the granularity of the classes. Too general a class will provide no discriminatory power while too fine-grained a class will offer little generalization and apply in only extremely few cases.

The absence of an attested high-quality set of semantic classes for this task makes discovering preferences difficult. Since many of the criteria for developing such a set are not even known, we decided to experiment with two very different sets of semantic classes, in the hope that in addition to learning semantic preferences, we might also uncover some clues for the eventual decisions about what makes good semantic classes in general.

Our first set of semantic classes was directly extracted from the output of the CBC clustering algorithm (Pantel and Lin 2002). We applied CBC to the TREC-9 and TREC-2002 (Aquaint) newswire collections consisting of over 600 million words. CBC generated 1628 noun concepts and these were used as our semantic classes for SPs.

Secondly, we extracted semantic classes from WordNet 2.1 (Fellbaum 1998). In the absence of any externally motivated distinguishing features (for example, the Basic Level categories from Prototype Theory, developed by Eleanor Rosch (1978)), we used the simple but effective method of manually truncating the noun synset hierarchy[5] and considering all synsets below each cut point as part of the semantic class at that node. To select the cut points, we inspected several different hierarchy levels and found the synsets at a depth of 4 to form the most natural semantic classes. Since the noun hierarchy in WordNet has an average depth of 12, our truncation created a set of concepts considerably coarser-grained than WordNet itself. The cut produced 1287 semantic classes, a number similar to the classes in CBC. To properly test WordNet as a source of semantic classes for our selectional preferences, we would need to experiment with different extraction algorithms.

4.3 Evaluation Criteria

The goal of the filtering task is to minimize false positives (incorrectly accepted inferences) and false negatives (incorrectly rejected inferences). A standard methodology for evaluating such tasks is to compare system filtering results with a gold standard using a confusion matrix. A confusion matrix captures the filtering performance on both correct and incorrect inferences:

		GOLD STANDARD	
		1	**0**
SYSTEM	**1**	*A*	*B*
	0	*C*	*D*

where *A* represents the number of correct instances correctly identified by the system, *D* represents the number of incorrect instances correctly identified by the system, *B* represents the number of false positives and *C* represents the number of false negatives. To compare systems, three key measures are used to summarize confusion matrices:

- ***Sensitivity***, defined as $\frac{A}{A+C}$, captures a filter's probability of accepting correct inferences;
- ***Specificity***, defined as $\frac{D}{B+D}$, captures a filter's probability of rejecting incorrect inferences;
- ***Accuracy***, defined as $\frac{A+D}{A+B+C+D}$, captures the probability of a filter being correct.

5 Experimental Results

In this section, we provide empirical evidence to support the main claim of this paper.

Given a collection of DIRT inference rules of the form $p_i \Rightarrow p_j$, our experiments, using the methodology of Section 4, evaluate the capability of our *ISP* models for determining if $\langle x, p_j, y\rangle$ holds given that $\langle x, p_i, y\rangle$ holds.

[5] Only nouns are considered since DIRT semantic relations connect only nouns.

Table 1. Filtering quality of best performing systems according to the evaluation criteria defined in Section 4.3 on the TEST set – the reported systems were selected based on the *Accuracy* criterion on the DEV set.

SYSTEM		PARAMETERS SELECTED FROM DEV SET		SENSITIVITY (95% CONF)	SPECIFICITY (95% CONF)	ACCURACY (95% CONF)
		RANKING STRATEGY	τ (%)			
B0		-	-	0.00±0.00	1.00±0.00	0.50±0.04
B1		-	-	1.00±0.00	0.00±0.00	0.49±0.04
Random		-	-	0.50±0.06	0.47±0.07	0.50±0.04
CBC	**ISP.JIM**	**maximum**	**100**	**0.17±0.04**	**0.88±0.04**	**0.53±0.04**
	ISP.IIM.$\wedge$	maximum	100	0.24±0.05	0.84±0.04	0.54±0.04
	ISP.IIM.$\vee$	**maximum**	**90**	**0.73±0.05**	**0.45±0.06**	**0.59±0.04**[†]
WordNet	ISP.JIM	minimum	40	0.20±0.06	0.75±0.06	0.47±0.04
	ISP.IIM.$\wedge$	minimum	10	0.33±0.07	0.77±0.06	0.55±0.04
	ISP.IIM.$\vee$	minimum	20	0.87±0.04	0.17±0.05	0.51±0.05

[†] Indicates statistically significant results (with 95% confidence) when compared with all baseline systems using pairwise *t*-test.

5.1 Experimental Setup

Model Implementation

For each filtering algorithm in Section 3.3, ISP.JIM, ISP.IIM.$\wedge$, and ISP.IIM.$\vee$, we trained their probabilistic models using corpus statistics extracted from the 1999 AP newswire collection (part of the TREC-2002 Aquaint collection) consisting of approximately 31 million words. We used the Minipar parser (Lin 1993) to match DIRT patterns in the text. This permits exact matches since DIRT inference rules are built from Minipar parse trees.

For each system, we experimented with the different ways of combining relational SP scores: *minimum*, *maximum*, and *average* (see Section 3.2). Also, we experimented with various values for the τ parameter described in Section 3.3.

Gold Standard Construction

In order to compute the confusion matrices described in Section 4.3, we must first construct a representative set of inferences and manually annotate them as correct or incorrect.

We randomly selected 100 inference rules of the form $p_i \Rightarrow p_j$ from DIRT. For each pattern p_i, we then extracted its instances from the Aquaint 1999 AP newswire collection (approximately 22 million words), and randomly selected 10 distinct instances, resulting in a total of 1000 instances. For each instance of p_i, applying DIRT's inference rule would assert the instance $\langle x, p_j, y\rangle$. Our evaluation tests how well our models can filter these so that only correct inferences are made.

To form the gold standard, two human judges were asked to tag each instance $\langle x, p_j, y\rangle$ as correct or incorrect. For example, given a randomly selected inference rule "*X is charged by Y* $\Rightarrow$ *Y announced the arrest of X*" and the instance "*Terry Nichols was charged by federal prosecutors*", the judges must determine if the instance $\langle$*federal prosecutors*, *Y announced the arrest of X*, *Terry Nichols*$\rangle$ is correct. The judges were asked to consider the following two criteria for their decision:

- $\langle x, p_j, y\rangle$ is a semantically meaningful instance;
- The inference $p_i \Rightarrow p_j$ holds for this instance.

Judges found that annotation decisions can range from trivial to difficult. The differences often were in the instances for which one of the judges fails to see the right context under which the inference could hold. To minimize disagreements, the judges went through an extensive round of training.

To that end, the 1000 instances $\langle x, p_j, y\rangle$ were split into DEV and TEST sets, 500 in each. The two judges trained themselves by annotating DEV together. The TEST set was then annotated separately to verify the inter-annotator agreement and to verify whether the task is well-defined. The kappa statistic (Siegel and Castellan Jr. 1988) was $\kappa = 0.72$. For the 70 disagreements between the judges, a third judge acted as an adjudicator.

Baselines

We compare our *ISP* algorithms to the following baselines:

- ***B0***: Rejects all inferences;
- ***B1***: Accepts all inferences;
- ***Rand***: Randomly accepts or rejects inferences.

One alternative to our approach is admit instances on the Web using literal search queries. We investigated this technique but discarded it due to subtle yet critical issues with pattern canonicalization that resulted in rejecting nearly all inferences. However, we are investigating other ways of using Web corpora for this task.

a)

		GOLD STANDARD	
		1	0
SYSTEM	1	184	139
	0	63	114

b)

		GOLD STANDARD	
		1	0
SYSTEM	1	42	28
	0	205	225

Figure 1. Confusion matrices for *a*) ISP.IIM.∨ – best *Accuracy*; and *b*) ISP.JIM – best *90%-Specificity*.

5.2 Filtering Quality

For each *ISP* algorithm and parameter combination, we constructed a confusion matrix on the development set and computed the system sensitivity, specificity and accuracy as described in Section 4.3. This resulted in 180 experiments on the development set. For each *ISP* algorithm and semantic class source, we selected the best parameter combinations according to the following criteria:

- *Accuracy*: This system has the best overall ability to correctly accept and reject inferences.
- *90%-Specificity*: Several formal semantics and textual entailment researchers have commented that inference rule collections like DIRT are difficult to use due to low precision. Many have asked for filtered versions that remove incorrect inferences even at the cost of removing correct inferences. In response, we show results for the system achieving the best sensitivity while maintaining at least 90% specificity on the DEV set.

We evaluated the selected systems on the TEST set. Table 1 summarizes the quality of the systems selected according to the *Accuracy* criterion. The best performing system, ISP.IIM.∨, performed statistically significantly better than all three baselines. The best system according to the *90%-Specificity* criteria was ISP.JIM, which coincidentally has the highest accuracy for that model as shown in Table 1[6]. This result is very promising for researchers that require highly accurate inference rules since they can use ISP.JIM and expect to recall 17% of the correct inferences by only accepting false positives 12% of the time.

Performance and Error Analysis

Figures 1a) and 1b) present the full confusion matrices for the most accurate and highly specific systems, with both systems selected on the DEV set. The most accurate system was ISP.IIM.∨, which is the most permissive of the algorithms. This suggests that a larger corpus for learning SPs may be needed to support stronger performance on the more restrictive methods. The system in Figure 1b), selected for maximizing sensitivity while maintaining high specificity, was 70% correct in predicting correct inferences.

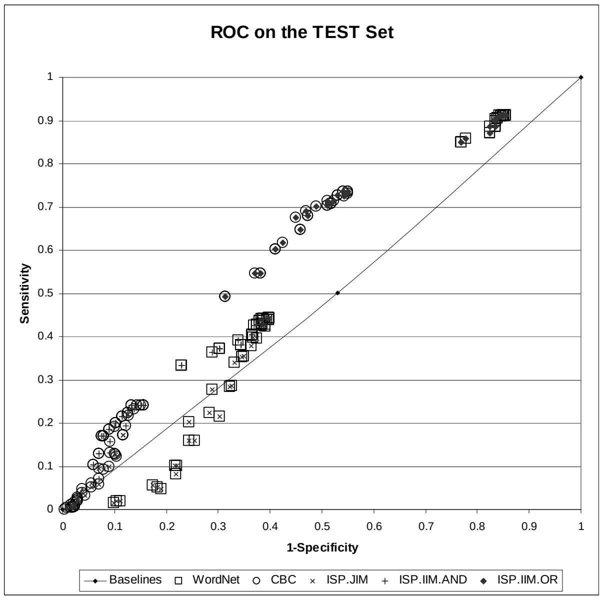


Figure 2. ROC curves for our systems on TEST.

Figure 2 illustrates the ROC curve for all our systems and parameter combinations on the TEST set. ROC curves plot the true positive rate against the false positive rate. The near-diagonal line plots the three baseline systems.

Several trends can be observed from this figure. First, systems using the semantic classes from WordNet tend to perform less well than systems using CBC classes. As discussed in Section 4.2, we used a very simplistic extraction of semantic classes from WordNet. The results in Figure 2 serve as a lower bound on what could be achieved with a better extraction from WordNet. Upon inspection of instances that WordNet got incorrect but CBC got correct, it seemed that CBC had a much higher lexical coverage than WordNet. For example, several of the instances contained proper names as either the *X* or *Y* argument (WordNet has poor proper name coverage). When an argument is not covered by any class, the inference is rejected.

Figure 2 also illustrates how our three different *ISP* algorithms behave. The strictest filters, ISP.JIM and ISP.IIM.∧, have the poorest overall performance but, as expected, have a generally very low rate of false positives. ISP.IIM.∨, which is a much more permissive filter because it does not require

[6] The reported sensitivity of *ISP.Joint* in Table 1 is below 90%, however it achieved 90.7% on the DEV set.

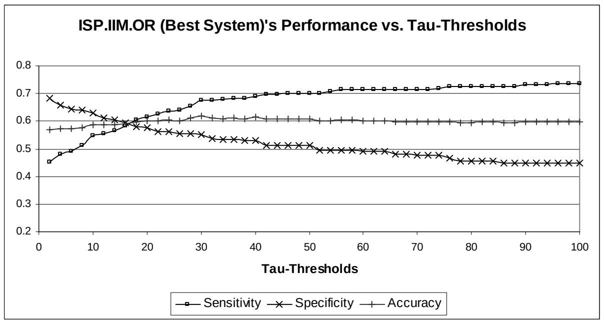


Figure 3. ISP.IIM.$\vee$ (Best System)'s performance variation over different values for the τ threshold.

both arguments of a relation to match, has generally many more false positives but has an overall better performance.

We did not include in Figure 2 an analysis of the *minimum*, *maximum*, and *average* ranking strategies presented in Section 3.2 since they generally produced nearly identical results.

For the most accurate system, ISP.IIM.$\vee$, we explored the impact of the cutoff threshold τ on the sensitivity, specificity, and accuracy, as shown in Figure 3. Rather than step the values by 10% as we did on the DEV set, here we stepped the threshold value by 2% on the TEST set. The more permissive values of τ increase sensitivity at the expense of specificity. Interestingly, the overall accuracy remained fairly constant across the entire range of τ, staying within 0.05 of the maximum of 0.62 achieved at τ=30%.

Finally, we manually inspected several incorrect inferences that were missed by our filters. A common source of errors was due to the many incorrect "antonymy" inference rules generated by DIRT, such as "*X is rejected in Y*"$\Rightarrow$"*X is accepted in Y*". This recognized problem in DIRT occurs because of the distributional hypothesis assumption used to form the inference rules. Our *ISP* algorithms suffer from a similar quandary since, typically, antonymous relations take the same sets of arguments for *X* (and *Y*). For these cases, *ISP* algorithms learn many selectional preferences that accept the same types of entities as those that made DIRT learn the inference rule in the first place, hence *ISP* will not filter out many incorrect inferences.

6 Conclusion

We presented algorithms for learning what we call inferential selectional preferences, and presented evidence that learning selectional preferences can be useful in filtering out incorrect inferences. Future work in this direction includes further exploration of the appropriate inventory of semantic classes used as SP's. This work constitutes a step towards better understanding of the interaction of selectional preferences and inferences, bridging these two aspects of semantics.

References

Barzilay, R.; and McKeown, K.R. 2001.Extracting Paraphrases from a Parallel Corpus. In *Proceedings of ACL 2001*. pp. 50–57. Toulose, France.

Baker, C.F.; Fillmore, C.J.; and Lowe, J.B. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING/ACL 1998*. pp. 86-90. Montreal, Canada.

Cover, T.M. and Thomas, J.A. 1991. *Elements of Information Theory*. John Wiley & Sons.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Harabagiu, S.; and Hickl, A. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of ACL 2006*. pp. 905-912. Sydney, Australia.

Katz, J.; and Fodor, J.A. 1963. The Structure of a Semantic Theory. *Language*, vol 39. pp.170–210.

Lenat, D. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Levin, B. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL.

Light, M. and Greiff, W.R. 2002. Statistical Models for the Induction and Use of Selectional Preferences. *Cognitive Science*,26:269–281.

Lin, D. 1993. Parsing Without OverGeneration. In *Proceedings of ACL-93*. pp. 112-120. Columbus, OH.

Lin, D. and Pantel, P. 2001. Discovery of Inference Rules for Question Answering. *Natural Language Engineering* 7(4):343-360.

Moldovan, D.I.; Clark, C.; Harabagiu, S.M.; Maiorano, S.J. 2003. COGEX: A Logic Prover for Question Answering. In *Proceedings of HLT-NAACL-03*. pp. 87-93. Edmonton, Canada.

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of KDD-02*. pp. 613-619. Edmonton, Canada.

Resnik, P. 1996. Selectional Constraints: An Information-Theoretic Model and its Computational Realization. *Cognition*, 61:127–159.

Romano, L.; Kouylekov, M.; Szpektor, I.; Dagan, I.; Lavelli, A. 2006. Investigating a Generic Paraphrase-Based Approach for Relation Extraction. In *EACL-2006*. pp. 409-416. Trento, Italy.

Rosch, E. 1978. Human Categorization. In E. Rosch and B.B. Lloyd (eds.) *Cognition and Categorization*. Hillsdale, NJ: Erlbaum.

Siegel, S. and Castellan Jr., N. J. 1988. Nonparametric Statistics for the Behavioral Sciences. McGraw-Hill.

Szpektor, I.; Tanev, H.; Dagan, I.; and Coppola, B. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*. pp. 41-48. Barcelona,Spain.

Wilks, Y.; and Fass, D. 1992. Preference Semantics: a family history. *Computing and Mathematics with Applications,* 23(2). A shorter version in the second edition of the Encyclopedia of Artificial Intelligence, (ed.) S. Shapiro.

Zanzotto, F.M.; Pennacchiotti, M.; Pazienza, M.T. 2006. Discovering Asymmetric Entailment Relations between Verbs using Selectional Preferences. In COLING/ACL-06. pp. 849-856. Sydney, Australia.

Computing Semantic Similarity between Skill Statements for Approximate Matching

Feng Pan
USC Information Sciences Institute
Marina del Rey, CA 90292
pan@isi.edu

Robert G. Farrell
IBM T. J. Watson Research Center
Hawthorne, NY 10532
robfarr@us.ibm.com

Abstract

This paper explores the problem of computing text similarity between verb phrases describing skilled human behavior for the purpose of finding approximate matches. Four parsers are evaluated on a large corpus of skill statements extracted from an enterprise-wide expertise taxonomy. A similarity measure utilizing common semantic role features extracted from parse trees was found superior to an information-theoretic measure of similarity and comparable to the level of human agreement.

1 Introduction

Knowledge-intensive industries need to become more efficient at deploying the right expertise as quickly and smoothly as possible, thus it is desired to have systems that can quickly match and deploy skilled individuals to meet customer needs. The searches in most of the current matching systems are based on *exact matches* between skill statements. However, exact matching is very likely to miss individuals who are very good matches to the job but didn't select the exact skills that appeared in the open job description.

It is always hard for individuals to find the perfect skills to describe their skill sets. For example, an individual might not know whether to choose a skill stating that refers to "maintaining" a given product or "supporting" it or whether to choose a skill about maintaining a "database" or about maintaining "DB2". Thus, it is desirable for the job search system to be able to find *approximate matches*, instead of only exact matches, between available individuals and open job positions. More specifically, a skill similarity computation is needed to allow searches to be expanded to related skills, and return more potential matches.

In this paper, we present our work on developing a skill similarity computation based upon semantic commonalities between skill statements. Although there has been much work on text similarity metrics (Lin, 1998a; Corley and Mihalcea, 2005), most approaches treat texts as a bag of words and try to find shared words with certain statistical properties based on corpus frequencies. As a result, the *structural information* in the text is ignored in these approaches. We will describe a new semantic approach that takes the structural information of the text into consideration and matches skill statements on corresponding *semantic roles*. We will demonstrate that it can outperform standard statistical text similarity techniques, and reach the level of human agreement.

In Section 2, we first describe the skill statements we extracted from an enterprise-wide expertise taxonomy. In Section 3, we describe the performance of a standard statistical approach on this task. This motivates our semantic approach of matching skill statements on corresponding semantic roles. We also compare and evaluate the performance of four natural language parsers (the Charniak parser, the Stanford parser, the ESG parser, and MINIPAR) for the purpose of our task. An inter-rater agreement study and evaluation of our approach will be presented in Section 4. We end with a discussion and conclusion.

2 Skill Statements

An expertise taxonomy is a standardized, enterprise-wide language and structure to describe job role requirements and people capabilities (skill sets) across a corporation. In the taxonomy we utilize for this study, skills are associated with job roles. The taxonomy has 10667 skills. Each skill has a title, for example, "Advise BAAN eBusiness ASP." We refer to this title as the *skill statement*.

The official taxonomy update policies require that skill statements be verb phrases using one of 18 valid skill verbs (e.g., *Advise, Architect, Code, Design, Implement, Sell, and Support*).

3 Computing Semantic Similarities between Skill Statements

In this section, we first explain a statistical information-theoretic approach we used as a baseline, and show examples of how it performs for our task. The error analysis of this approach motivates our semantic approach that takes the structural information of the text into consideration. In the remainder of this section, we describe how we extract semantic role information from the syntactic parse trees of the skill statements. Four natural language parsers are compared and evaluated for the purpose of our task.

3.1 Statistical Approach

In order to compute semantic similarities between skill statements, we first adopted one of the standard statistical approaches to the problem of computing text similarities based on Lin's information-theoretic similarity measure (Lin 1998a). Lin defined the commonality between A and B as

$$I(common(A,B))$$

where *common*(*A*, *B*) is a proportion that states the commonalities between A and B and where the amount of information in proposition *s* is

$$I(s) = -\log P(s)$$

The similarity between A and B is then defined as the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe A and B:

$$Sim(A,B) = \frac{\log P(common(A,B))}{\log P(description(A,B))}$$

In order to compute *common*(*A*,*B*) and *description*(*A*,*B*), we use standard bag-of-words features, i.e., unigram features -- the frequency of words computed from the entire corpus of the skill statements. Thus *common*(*A*,*B*) is the unigrams that both skill statements share, and *description*(*A*,*B*) is the union of the unigrams from both skill statements.

The words are stemmed first so that the words with the same root (e.g., *managing* & *management*) can be found as commonalities between two skill statements. A stop-word list is also used so that the commonly used words in most of the documents (e.g., *the*, *a*) are not used as features. A formal evaluation of this approach will be presented in Section 4 where the similarity between 75 pairs of skill statements will be evaluated against human judgments, but we discuss some examples here.

In order to see how to improve Lin's statistical similarity measure, we examine sample skill statement pairs which achieve high similarity scores from Lin's measure but were rated consistently as dissimilar by human subjects in our evaluation. Here are two examples:

1. Advise Business Knowledge of *CAD functionality* for FEM
   Advise on Business Knowledge of *Process* for FEM
2. Advise on *Money Market*
   Advise on *Money Center Banking*

In these two examples, although many words are shared between the two pairs of skill statements (*Advise Business Knowledge of ... for FEM* for the first pair; *Advise on Money* for the second pair), they are not similar to human judges. We conjecture that this judgment of dissimilarity is due to the differences between the key components of the skill statements (*CAD functionality* vs. *Process* in the first pair; *Money Market* vs. *Money Center Banking* in the second pair).

This kind of error is common for most statistical approaches to the problem, where common information is computed without considering the structural information in the text. From the above examples, we can see that the similarity computation would be more accurate if the verb phrases *match on corresponding semantic roles,* instead of matching words from any location in the skill statements. By identifying semantic roles, we can provide more weights to those semantic roles critical for our task, i.e., the key components of the skill statements.

3.2 Identifying and Assigning Semantic Roles

The following example shows the kind of semantic roles we want to be able to identify and assign.

> [$_{\textbf{action}}$ Apply] [$_{\textbf{theme}}$ Knowledge of [$_{\textbf{concept}}$ IBM E-business Middleware]] to [$_{\textbf{purpose}}$ PLM Solutions]

In this example, “Apply” is the “action” of the skill; “Knowledge of IBM E-business Middleware” is the “theme” of the skill, where the “concept” semantic role (*IBM E-business Middleware*) specifies the key component of the skill requirement and is the most important role for skill matching; “PLM Solutions” is the “purpose” of the skill.

Our goal was to extract all such semantic role patterns for all the skill statements, and match on corresponding semantic roles. Although there exists some automatic semantic role taggers (Gildea and Jurafsky, 2002; Giuglea and Moschitti, 2006), most of them were trained on PropBank (Palmer et. al., 2005) and/or FrameNet (Johnson et. al., 2003), and perform much worse in other corpora (Pradhan et. al., 2004). Our corpus is from a very different domain (information technology) and there are many domain-specific terms in the skill statements, such as product names, company names, and company-specific nomenclature for product offerings. Given this, we would expect poor performance from these automatic semantic role taggers. Moreover, the semantic role information we need to extract is more detailed and deeper than most of the automatic semantic role taggers can identify and extract (e.g., the “concept” role embedded within the “theme” role).

We developed a specialized parser that extracts semantic role patterns from each of the 18 skill verbs. This semantic role parser can achieve a much higher performance than the general-purpose semantic role taggers. The inputs needed for the semantic role parser are syntactic parse trees generated by a natural language parse of the original skill statements.

3.3 Preprocessing for Parsing

We first used the Charniak parser (2000) to parse the original skill statements. However, among all the 10667 skill statements, 1217 were not parsed as verb phrases, leading to very poor performance. After examining the error cases, we found that abbreviations are used widely in the skill statements. For example,

> Advise Solns Supp Bus Proc Reeng for E&E Eng Procs

These abbreviations made the system unable to determine the part of speech of some words, resulting in incorrect parses. Thus, the first step of the preprocessing was to expand abbreviations.

There were 225 valid abbreviations already identified by the expertise taxonomy team. However, we found many abbreviations that appeared in the skill statements but were not listed there. Since most abbreviations are not words found in a dictionary, in order to find the abbreviations that appear frequently in the skill statements, we first found all the words in the skill statements that were not in WordNet (Miller, 1990). We then ranked them based on their frequencies, and manually identified high frequency abbreviations. Using this approach, we added another 187 abbreviations to the list (a total of 412).

From the error cases, we also found that many words were mistagged as proper nouns, For example, “Technically” in

> Advise Technically for Simulation

was parsed as a proper noun. We realized the reason for this error was that all the words, except for prepositions, are capitalized in the original statements and the parser tends to tag them as proper nouns. To solve this problem, we changed all the capitalized words to lower case, except for the first word and the acronyms (words that have all letters capitalized, e.g., IBM). After applying these two steps of preprocessing, we parsed the skill statements again. This time, more than 200 additional skill statements were parsed as verb phrases after the preprocessing.

When we examined the error cases more closely, we found the errors occur mostly when the skill verbs can be both a noun and a verb (e.g., *design*, *plan*). In those cases, the parser may parse the entire statement as one noun phrase, instead of a verb phrase. In order to disambiguate such cases, we added a subject ("Employees") to all the skill statements to convert them into full sentences. After applying this additional step of preprocessing, we parsed the skill statements again. This time, only 28 skill statements were not parsed as sentences containing verb phrases, a significant improvement. The remaining errors were due to the use of some words as skill verbs, e.g., "architect"[1], not recognized as verbs by the parser.

3.4 Parser Evaluation and Comparison

While the Charniak parser performed well in our initial verb phrase (VP) test, we decided to compare the Charniak parser's performance with other parsers. For this evaluation, we compared it with the Stanford parser, the ESG parser, and MINIPAR.

The **Stanford parser** (Klein and Manning, 2003) is an unlexicalized statistical syntactic parser that was trained on the same corpus as the Charniak parser (the Penn TreeBank). Its parse tree has the same structure as the Charniak parser.

The **ESG** (English Slot Grammar) **parser** (McCord, 1980) is a rule-based parser based on the slot grammar where each phrase has a head and dependent elements, and is also marked with a syntactic role.

MINIPAR (Lin, 1998b), as a dependency parser, is very similar to the ESG parser in terms of its output. It represents sentence structures as a set of dependency relationships between head words.

Since our purpose is to use the syntactic parses as inputs to extract semantic role patterns, the correctness of the bracketing of the parses and the syntactic labels of the phrases (e.g., NP, VP, and PP) are the most important information for our purposes, whereas the POS (Part-Of-Speech) labels of individual words (e.g., nouns vs. proper nouns) are not that important (also, there are too many domain-specific terms in our data). Thus, our evaluation of the parses is only on the correctness of the bracketing and the syntactic labels of the phrases, not the correctness of the entire parse. For our task, the correctness of the prepositional phrase attachment is especially important for extracting accurate semantic role patterns (Gildea and Jurafsky, 2002). For example, for the sentence

Apply Knowledge of IBM E-business Middleware to PLM Solutions.

the correct bracketing should be

Apply [Knowledge [of [IBM E-business Middleware]]] [to [PLM Solutions]].

Thus the parser needs to correctly attach "of IBM E-business Middleware" to "Knowledge" and attach "to PLM Solutions" to "Apply", not "Knowledge".

To evaluate the performance of the parsers, we randomly picked 100 skill statements from our corpus, preprocessed them, and then parsed them using the four different parsers. We then evaluated the parses using the above evaluation measures. The parses were rated as correct or incorrect. No partial score was given. Figure 1 shows the evaluation results. The error analysis reveals four major sources of error for all the parsers, most of which are specific to the domain we are working on:

(1) Many domain specific terms and acronyms. For example, "SAP" in "Employees advise on *SAP* R/3 logistics basic data." was always tagged as a verb by the parsers.
(2) Many long noun phrases. For example, "Employees perform *JD edwards foundation suite address book*."
(3) Some specialized use of punctuation. For example, "Employees perform business transportation consultant-logistics.sys."
(4) Prepositional phrase attachment can be difficult. For example, in "Employees apply IBM infrastructure knowledge *for IDBS*", "for IDBS" should attach to "apply", but many parsers mistakenly attach it to "IBM infrastructure knowledge".

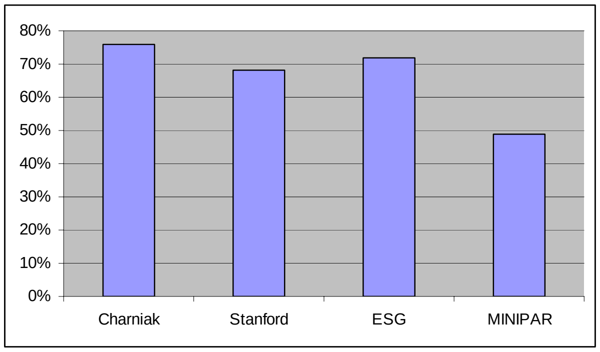


Figure 1. An Evaluation of Four Parsers on the Task of Parsing Human Skill-related Verb Phrases

We noticed that MINIPAR performed much worse compared with the other parsers. The main

[1] "Architect" has no verb sense in WordNet and many other dictionaries, but it does have a verb sense in the Oxford English Dictionary (http://dictionary.oed.com/).

reason is that it always parses the phrase "VERB knowledge of Y" (e.g., "Employees *apply knowledge of web technologies*.") incorrectly -- the parse result always mistakenly attaches "of Y" (e.g., *of web technologies*) to the VERB (e.g., *apply*), not "knowledge". Since there were so many *of* phrases in the test set and in the corpus, this kind of error significantly reduced the performance for our task. These kinds of errors on prepositional phrase attachment in MINIPAR were also mentioned in (Pantel and Lin, 2000).

From the evaluation and comparison results we can see that the Charniak parser performs the best for our task among all the four parsers. This result is consistent with a more thorough evaluation (Swanson and Gordon, 2006) on a different corpus with a set of different target verbs, which showed the Charniak parser performed the best among three parsers (including the Stanford parser and MINPAR) for labeling semantic roles. We note that although the ESG parser performed a little worse than the Charniak parser, its parses contain much richer syntactic (e.g., subject, object) and semantic (e.g., word senses) slot-filling information, which can be very useful to many natural language applications.

3.5 Extracted Semantic Role Patterns

From the parse trees generated by the Charniak parser, we first automatically extracted patterns for each of the 18 skill verbs (e.g., "Advise on NP for NP"), and then we manually identified the semantic roles. For example, the semantic role patterns identified for the skill verb "Advise" are:

- Advise [Theme] (for [Purpose])
- Advise (technically) on/about [Theme] (for [Purpose])
- Advise clients/customers/employees/users on/regarding [Theme]

The corpus also contains embedded sub-semantic-role patterns, for example, for the "Theme" role we extracted the following sub-patterns:

- (application) knowledge of/for [Concept]
- sales of [Concept]
- (technical) implementation of [Concept]

We have extracted and identified a total of 74 such semantic role patterns from the skill statements.

4 Evaluation

In order to evaluate the two approaches (semantic role parsing and statistical) to computing semantic similarity of skill statements in our domain, we first conducted an experiment to evaluate how humans agree on this task, which also provides us with an upper bound accuracy for the task.

4.1 Inter-Rater Agreement and Upper Bound Accuracy

To assess inter-rater agreement, we randomly selected 75 skill pairs from the expertise taxonomy. Since random pairs of verbs would have little or no similarity, we selected skill pairs that share the same job role, or same secondary or primary job category, or from across the entire expertise taxonomy.

These 75 skill pairs are then given to three raters to independently judge their similarities on a 5 point scale from 1 as very similar to 5 as very dissimilar. Since this 5 point scale is very fine-grained, we also converted the judgments to a more coarse-grained measure -- binary judgment: 1 and 2 count as similar; 3-5 as not similar.

The metric we used is the kappa statistic (Carletta, 1996), which factors out the agreement that is expected by chance:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

where $P(A)$ is the observed agreement among the raters, and $P(E)$ is the expected agreement, i.e., the probability that the raters agree by chance.

Since the judgment on the 5 point scale is ordinal data, the weighted kappa statistic is used to take the distance of disagreement into consideration (e.g., the disagreement between 1 and 2 is smaller than that between 1 and 5).

The inter-rater agreement results for both the fine-grained and coarse-grained judgments are shown in Table 1. In general, a kappa value above 0.80 represents perfect agreement, 0.60-0.80 represents significant agreement, 0.40-0.60 represents moderate agreement, and 0.20-0.40 is fair agreement (Chklovski and Mihalcea, 2003). We can see that the agreement on the fine-grained judgment is moderate, whereas the agreement on the coarse-grained (binary) judgment is significant.

	Fine-Grained	Coarse-Grained
Kappa	0.412	0.602

Table 1. Inter-Rater Agreement Results.

From the inter-rater agreement evaluation, we can also get an *upper bound accuracy* for our task, i.e., human agreement without factoring out the agreement expected by chance (i.e., $P(A)$ in the kappa statistic). The average $P(A)$ for the coarse-grained (binary) judgment is 0.81, and that constitutes the upper bound accuracy for our task.

4.2 Evaluation of the Statistical Approach

We use the 75 skill pairs as test data to evaluate our semantic similarity approach against human judgments. Considering the reliability of the data, only the coarse-grained (binary) judgments are used. The gold standard is obtained by majority voting from the three raters, i.e., for a given skill pair, if two or more raters judge it as similar, then the gold standard answer is "similar", otherwise it is "not similar".

We first evaluated Lin's statistical approach described in Section 3.1. Among 75 skill pairs, 53 of them were rated correctly according to the human judgments, that is, 70.67% accuracy. The error analysis shows that many of the errors can be corrected if the skills are matched on their corresponding semantic roles. We then evaluated the utility of the extracted semantic role information to see whether it can outperform the statistical approach.

4.3 Evaluation of Semantic Role Matching Approach

For simplicity, we will only report on evaluating semantic role matching on the "concept" role that specifies the key component of the skills, as introduced in Section 3.2.

There are at least two straightforward ways of performing semantic role matching for the skill similarity computation: 1) match on the entire semantic role; 2) match on the head nouns only. But both have their drawbacks: the first approach is too strict and will miss many similar skill statements; the second approach may not only miss the similar skill statements, e.g.,

Perform [Web Services *Planning*][2]
Perform [Web Services *Assessment*]

but also misclassify dissimilar ones as similar, e.g.,

Advise about [Async Transfer Mode (ATM) *Solutions*]
Advise about [CTI *Solutions*]

In order to solve these problems, we used a simple matching criterion from Tversky (1977). The similarity of two texts t_1 and t_2 is determined by:

$$\text{Similarity}(t_1, t_2) = \frac{2 \times (\#\,\text{common features between } t_1 \text{ and } t_2)}{\#\,\text{total features in } t_1 \text{ and } t_2}$$

This equation states that *two texts are similar if shared features are a large percentage of the total features.* We set a threshold of 0.5, requiring that at least 50% of the features be shared. We apply this criterion to the text contained in the "concept" role.

The words in the calculation are preprocessed first: abbreviations are expanded, stop-words are excluded (e.g., *the* and *of* don't count as shared words), and the remaining words are stemmed (e.g., *manager* and *management* are counted as shared words), as was done in our previous information-theoretic approach. Words connected by punctuation (e.g., *e-business*, *software/hardware*) are treated as separate words. For example,

Advise on [*Field*/*Force* Management] for Telecom
Apply Knowledge of [Basic *Field Force* Automation]

The shared words between the two "concept" roles (bracketed) are "Field" and "Force", and their shared percentage is (2*2)/7 = 57.14% > 50%, so they are similar.

We have also evaluated this approach on our test set with the 75 skill pairs. Among 75 skill pairs, 60 of them were rated correctly (i.e., 80% accuracy), which significantly outperforms the statistical approach, and is very close to the upper bound accuracy, i.e., human agreement (81%), as shown in Figure 2.

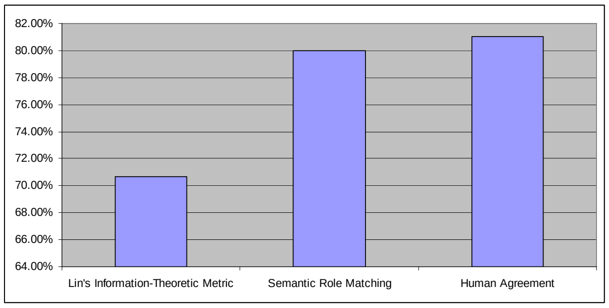


Figure 2. Evaluation on Semantic Similarity between Skill Statements

[2] The "concept" role is identified with brackets, and the head nouns are italic.

The difference between this approach and Lin's information content approach is that this computation is local -- no corpus statistics is used. Also, using this approach, it is easier to set an intuitive threshold (e.g., 50%) for a classification problem (e.g., *similar or not* for our task). With this approach, however, there are also cases that are mistagged as similar, for example,

Apply Knowledge of [Basic Field *Force Automation*]
Advise on [Sales *Force Automation*]

Although "Field Force Automation" and "Sales Force Automation" seem similar on their surface form, they are two quite different concepts. Deeper domain knowledge (such as an ontology) is needed to distinguish such cases.

5 Discussion

We have also investigated several approaches to improving the semantic role text similarity measure we described. One approach is to also consider similarities between skill verbs. In this example:

Implement *Domino Mail Manager*
Develop for *Domino Mail Manager*

although the key components of the skill statements (*Domino Mail Manager*) are the same, their skill verbs are different (*implement* vs. *develop for*). The skills required for "implementing" a system or software product are usually different from those required for "developing for" the same system or software product. This example shows that a semantic similarity computation between skill verbs is required to distinguishing such cases.

Many approaches to the problem of word/concept similarities are based on taxonomies, e.g., WordNet. The simplest approach is to count the number of nodes on the shortest path between two concepts in the taxonomy (Quillian, 1972). The fewer nodes on the path, the more similar the two concepts are. The assumption for this shortest path approach is that the links in the taxonomy represent uniform distances. However, in most taxonomies, sibling concepts deep in the taxonomy are usually more closely related than those higher up. Different approaches have been proposed to discount the depth of the concepts to overcome the problem. Budanitsky and Hirst (2006) thoroughly evaluated six of the approaches (Hirst and St-Onge, Leacock and Chodorow, Jiang and Conrath, Lin, Resnik, Wu and Palmer), and found that Jiang and Conrath (1997) was superior to the other approaches based on their evaluation experiments.

For our task, we compared two approaches to computing skill verb similarities: shortest path vs. Jiang and Conrath. Since the words are compared based on their specific senses, we first manually assigned one most appropriate sense for each of the 18 skill verbs from WordNet. We then used the library developed by Pedersen et al. (2004) to compute their similarity scores.

Table 2 shows the top nine pairs of skill verbs with the highest similarity scores from the two approaches. We can see that the two approaches agree on the top four pairs, but disagree on the rest in the list. One intuitive example is the pair "Lead" and "Manage" which is ranked the 5th by the Jiang and Conrath approach but ranked the 46th by the shortest path approach. It seems that the Jiang and Conrath approach matches better with our human intuition for this example. While we didn't compare these results with human performance, in general most of the similar skill verb pairs listed in the table don't look very similar for our domain. This may be due to the fact that WordNet is a general-purpose taxonomy -- although we have already selected the most appropriate sense for each verb, their relationship represented in the taxonomy may still be quite different from the relationship in our domain. A domain-specific taxonomy for skill verbs may improve the performance. The other reason may be due to the structure of WordNet's verb taxonomy, as mentioned in (Resnik and Diab, 2000), which is considerably wider and shallower than WordNet's noun taxonomy. A different verb lexicon, e.g., VerbNet (Kipper et al., 2000), can be explored.

Shortest Path		**Jiang and Conrath**	
Apply	Use	Apply	Use
Design	Plan	Design	Plan
Apply	Implement	Apply	Implement
Implement	Use	Implement	Use
Analyze	Apply	**Lead**	**Manage**
Analyze	Perform	Apply	Support
Analyze	Support	Support	Use
Analyze	Use	Apply	Sell
Perform	Support	Sell	Use
…	**…**	**…**	**…**

Table 2. Top Similar Skill Verb Pairs

6 Conclusion

In this paper, we have presented our work on a semantic similarity computation for skill statements in natural language. We compared and evaluated four different natural language parsers for our task, and matched skills on their corresponding semantic roles extracted from the parse trees generated by one of these parsers. The evaluation results showed that the skill similarity computation based on semantic role matching can outperform a standard statistical approach and reach the level of human agreement.

The extracted semantic role information can also be incorporated into the standard statistical approaches as additional features. One way is to give higher weights to those semantic role features deemed most important. This approach has achieved a high performance for a text categorization task when combining extracted keywords with the full text (Hulth and Megyesi, 2006).

We have shown that good results can be achieved for a domain-specific text matching task by performing a simple word-based feature comparison on corresponding structural elements of texts. We have shown that the structural elements of importance can be identified by domain-specific pattern analysis of corresponding parse trees. We believe this approach can generalize to other domains where phrases, sentences, or other short texts need to be compared.

Acknowledgements

The majority of this work was performed while the first author was a summer intern at IBM T. J. Watson Research Center in Hawthorne, NY. Thanks to Yael Ravin and Jennifer Lai for supporting this work, Brian White for his help on the software, Michael McCord for assistance with the IBM ESG parser, and the IBM Expertise Taxonomy team for letting us use their data.

References

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*. 32(1):13-47.

J. Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254.

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

T. Chklovski and R. Mihalcea. 2003. Exploiting Agreement and Disagreement of Human Annotators for Word Sense Disambiguation. In *Proceedings of RANLP*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3): 245 – 288.

A. Giuglea and A. Moschitti. 2006. Semantic Role Labeling via FrameNet, VerbNet and PropBank. In *Proceedings of COLING-ACL*.

A. Hulth and B. B. Megyesi. 2006. A Study on Automatically Extracted Keywords in Text Categorization. In *Proceedings of COLING-ACL*.

J. J. Jiang and D. W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*.

C. Johnson, M. Petruck, C. Baker, M. Ellsworth, J. Ruppenhofer, and C. Fillmore. 2003. *Framenet: Theory and practice*. Berkeley, California.

K. Kipper, H. T. Dang, M. Palmer. 2000. Class-Based Construction of a Verb Lexicon. In *Proceedings of AAAI*.

D. Klein and C. D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*.

D. Lin. 1998a. An information-theoretic definition of similarity. In *Proceedings of ICML*.

D. Lin. 1998b. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop at LREC on The Evaluation of Parsing Systems*.

M. C. McCord. 1980. Slot grammars. *Computational Linguistics*, 6: 31-43.

G. A. Miller. 1990. WordNet: an On-line Lexical Database. *International Journal of Lexicography 3(4)*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics,* 31(1).

P. Pantel and D. Lin. 2000. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *Proceedings of ACL*.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of AAAI*, Intelligent Systems Demonstration.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow Semantic Parsing using Support Vector Machines. In *Proceedings of HLT/NAACL*.

M. R. Quillian. 1972. Semantic Memory, Semantic Information Processing. *Semantic information processing*, Cambridge.

P. Resnik and M. Diab. 2000. Measuring verb similarity. In *Proceedings of COGSCI*.

R. Swanson and A. S. Gordon. 2006. A Comparison of Alternative Parse Tree Paths for Labeling Semantic Roles. In *Proceedings of COLING/ACL*.

A. Tversky. 1977. Features of Similarity, *Psychological Review*, vol. 84, no. 4, pages 327-352.

Author Index